IMA workshop – Optimization in simulation based models

Generalized Pattern Search Algorithms:
unconstrained and constrained cases

Mark A. Abramson
Air Force Institute of Technology

Charles Audet
École Polytechnique de Montréal

John Dennis
Rice University

www.gerad.ca/Charles.Audet/
Generalized Pattern Search (GPS)

- Unconstrained problem
  \[
  \min_{x \in \mathbb{R}^n} f(x)
  \]
Presentation outline

Generalized Pattern Search (GPS)

♦ Unconstrained problem
\[
\min_{x \in \mathbb{R}^n} f(x)
\]

♦ Bound and linear constraints
\[
\min_{x \in X} f(x) \text{ where } X = \{x \in \mathbb{R}^n : Ax \leq b\} \cap [\ell, u]
\]
Presentation outline

Generalized Pattern Search (GPS)

♦ Unconstrained problem
\[
\min_{x \in \mathbb{R}^n} f(x)
\]

♦ Bound and linear constraints
\[
\min_{x \in X} f(x) \text{ where } X = \{x \in \mathbb{R}^n : Ax \leq b\} \cap [\ell, u]
\]

♦ General constraints
\[
\min_{x \in X \cap \Omega} f(x) \text{ where } \Omega = \{x : c_i(x) \leq 0, i = 1, 2, \ldots, m\}
\]
Unconstrained optimization

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

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\} \) may be discontinuous or infinite valued, and:

- \( f \) is usually given as a black box (typically a computer code),
- \( f \) is expensive and have few correct digits,
- \( f(x) \) may fail expensively and unexpectedly.
Ancestor of GPS : Coordinate search

♦ Initialization:

$x_0$ : initial point in $\mathbb{R}^n$

$\Delta_0 > 0$ : initial step size.
Ancestor of GPS: Coordinate search

♦ Initialization:

- $x_0$: initial point in $\mathbb{R}^n$
- $\Delta_0 > 0$: initial step size.

♦ Poll step: For $k = 0, 1, \ldots$

If $f(t) < f(x_k)$ for some $t \in P_k := \{x_k \pm \Delta_k e_i : i \in N\}$,
set $x_{k+1} = t$
and $\Delta_{k+1} = \Delta_k$;
Ancestor of GPS: Coordinate search

♦ Initialization:
  \( x_0 \) : initial point in \( \mathbb{R}^n \)
  \( \Delta_0 > 0 \) : initial step size.

♦ Poll step: For \( k = 0, 1, \ldots \)
  If \( f(t) < f(x_k) \) for some \( t \in P_k := \{ x_k \pm \Delta_k e_i : i \in N \} \),
  set \( x_{k+1} = t \)
  and \( \Delta_{k+1} = \Delta_k \);
  otherwise \( x_k \) is a minimizer over the set \( P_k \),
  set \( x_{k+1} = x_k \)
  and \( \Delta_{k+1} = \frac{\Delta_k}{2} \).
Coordinate search run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4401 \]
## Coordinate search run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[
\begin{array}{cc}
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\quad & \quad \\
\end{array}
\]

\[ f = 4401 \quad f = 29286 \]
Coordinate search run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4772 \]

\[ f = 4401 \quad f = 29286 \]
Coordinate search run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4772 \quad f = 166 \quad f = 4401 \quad f = 29286 \]
Coordinate search run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4772 \quad f = 4401 \quad f = 29286 \]

\[ f = 4176 \]

IMA, Jan 2003
Coordinate search run

\[ x_1 = (1,2)^T, \Delta_1 = 1 \]

\[ f = 166 \]
Coordinate search run

\[ x_1 = (1,2)^T, \Delta_1 = 1 \]

\[ f = 262 \]

\[ f = 81 \]

\[ f = 166 \]

\[ f = 4401 \]

\[ f = 106 \]
Coordinate search run

\[ x_2 = (2,2)^T, \Delta_2 = 1 \]

\[ f = 81 \]
Coordinate search run

\[ x_2 = (2, 2)^T, \Delta_2 = 1 \]

\[ f = 2646 \quad f = 81 \quad f = 166 \]

\[ f = 152 \quad f = 36 \]
Coordinate search run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]

\[ f = 36 \]
Coordinate search run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]

\[ f = 2466 \quad f = 36 \quad f = 106 \quad f = 81 \quad f = 17 \]
Coordinate search run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]

\[ f = 17 \]
Coordinate search run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]
Coordinate search run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]

\[ x_4 \] is called a mesh local optimizer

\[ f = 2402 \quad f = 17 \quad f = 82 \]

\[ f = 24 \]
Coordinate search run

\[ x_5 = x_4 = (0, 0)^T, \Delta_4 = \frac{1}{2} \]

\[ f = 17 \]
Coordinate search run

A budget of 20 function evaluations produces

\[ x = (0, 0)^T \text{ with } f(x) = 17. \]

Can we do better with the coordinate search?
Coordinate search : opportunistic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4401 \]
Coordinate search: opportunistic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ \begin{array}{cc}
  f = 4401 & f = 29286 \\
\end{array} \]
Coordinate search: opportunistic run

\[ x_0 = (2,2)^T, \Delta_0 = 1 \]

\[ f = 4772 \]

\[ f = 4401 \quad f = 29286 \]
Coordinate search : opportunistic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4772 \]

\[ f = 4401 \]

\[ f = 29286 \]
Coordinate search: opportunistic run

\[ x_1 = (1, 2)^T, \Delta_1 = 1 \]

\[ f = 166 \]
Coordinate search: opportunistic run

\[ x_1 = (1, 2)^T, \Delta_1 = 1 \]

\[ f = 262 \]

\[ f = 81 \]

\[ f = 166 \]

\[ f = 4401 \]
Coordinate search : opportunistic run

\[ x_2 = (2,2)^T, \Delta_2 = 1 \]

\[ f = 81 \]
Coordinate search : opportunistic run

\[ x_2 = (2, 2)^T, \Delta_2 = 1 \]

\[ f = 2646 \quad f = 81 \quad f = 166 \]

\[ f = 152 \]

\[ f = 36 \]
Coordinate search: opportunistic run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]

\[ f = 36 \]
Coordinate search: opportunistic run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]

\[ f = 81 \]

\[ f = 2466 \]

\[ f = 36 \]

\[ f = 106 \]

\[ f = 17 \]
Coordinate search: opportunistic run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]
Coordinate search: opportunistic run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]
Coordinate search: opportunistic run

\[ x_5 = x_4 = (0, 0)^T, \Delta_4 = \frac{1}{2} \]

\[ f = 17 \]
Coordinate search : opportunistic run

\[ x_5 = x_4 = (0,0)^T, \Delta_4 = \frac{1}{2} \]

\[ f = 17 \quad f = 1.0625 \]
Coordinate search : opportunistic run

\[ x_6 = (0, 0.5)^T, \Delta_4 = \frac{1}{2} \]
Coordinate search: opportunistic run

A budget of 20 function evaluations produces

\[ x = (0.5, 0)^T \text{ with } f(x) = 1.0625. \]

Can we do better with the coordinate search?
Coordinate search: dynamic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4401 \]
Coordinate search: dynamic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4401 \quad f = 29286 \]
Coordinate search : dynamic run

\( x_0 = (2, 2)^T, \Delta_0 = 1 \)

\( f = 4772 \)

\( f = 4401 \quad f = 29286 \)
Coordinate search: dynamic run

\[ x_0 = (2, 2)^T, \Delta_0 = 1 \]

\[ f = 4772 \]

\[ f = 166 \]

\[ f = 4401 \]

\[ f = 29286 \]
 Coordinate search : dynamic run

\[ x_0 = (2,2)^T, \Delta_0 = 1 \]

\[ f = 4772 \]

\[ f = 166 \]

\[ f = 4401 \]

\[ f = 29286 \]
Coordinate search: dynamic run

\[ x_1 = (1, 2)^T, \Delta_1 = 1 \]

\[ f = 166 \]
Coordinate search: dynamic run

\[ x_1 = (1,2)^T, \Delta_1 = 1 \]

\[ f = 81 \quad f = 166 \]
Coordinate search : dynamic run

\[ x_2 = (2, 2)^T, \Delta_2 = 1 \]

\[ f = 81 \]
Coordinate search : dynamic run

\[ x_2 = (2, 2)^T, \Delta_2 = 1 \]

\[ f = 2646 \quad f = 81 \]
Coordinate search: dynamic run

\[ x_2 = (2, 2)^T, \Delta_2 = 1 \]

| \( f = 2646 \) | \( f = 81 \) | \( f = 166 \) |
Coordinate search : dynamic run

$x_2 = (2,2)^T, \Delta_2 = 1$

\[ f = 2646 \quad f = 81 \quad f = 166 \quad f = 152 \]
Coordinate search : dynamic run

\[ x_2 = (2,2)^T, \Delta_2 = 1 \]

- \( f = 2646 \)
- \( f = 81 \)
- \( f = 166 \)
- \( f = 36 \)
Coordinate search : dynamic run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]

\[ f = 36 \]
Coordinate search : dynamic run

\[ x_3 = (0,1)^T, \Delta_3 = 1 \]
Coordinate search: dynamic run

\[ x_4 = (0,0)^T, \Delta_4 = 1 \]
Coordinate search: dynamic run

\[ x_4 = (0, 0)^T, \Delta_4 = 1 \]

\[ f = 24 \]
\[ f = 17 \]
\[ f = 36 \]
\[ f = 82 \]
\[ f = 2402 \]
Coordinate search : dynamic run

\[ x_5 = x_4 = (0,0)^T, \Delta_4 = \frac{1}{2} \]

\[ f = 17 \]
Coordinate search : dynamic run

\[ x_5 = x_4 = (0,0)^T, \Delta_4 = \frac{1}{2} \]

\[ f > 411, \quad f = 17, \quad f = 1.0625, \quad f = 17.25 \]
Coordinate search : dynamic run

\[ x_6 = (0, 0.5)^T, \Delta_4 = \frac{1}{2} \]

\[ f = 1.0625 \]
Coordinate search: dynamic run

\[ x_6 = (0, 0.5)^T, \Delta_4 = \frac{1}{2} \]
### Coordinate search: 3 strategies

<table>
<thead>
<tr>
<th>Complete</th>
<th>Fixed order</th>
<th>Dynamic order</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^T f(x)$</td>
<td>$x^T f(x)$</td>
<td>$x^T f(x)$</td>
</tr>
</tbody>
</table>

After 20 function evaluations:

- ($0, 0$) $17$
- ($0.5, 0$) $1.0625$
- ($0.5, -0.5$) $0.375$

After 50 function evaluations:

- ($0.375, -0.375$) $1.8e-2$
- ($0.375, -0.312$) $5.7e-3$
- ($0.375, -0.344$) $3.1e-3$

IMA, Jan 2003
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[\lim_{k} \Delta_k = 0\]

there is an \( \hat{x} \) which is the limit of a sequence of mesh local optimizers

If \( f \) is continuously differentiable at \( \hat{x} \), then \( \nabla f(\hat{x}) = 0 \).
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[ \lim_{k \to \infty} \Delta_k = 0 \]

\[ \exists \hat{x} \text{ which is the limit of a sequence of mesh local optimizers} \]

\[ \text{If } f \text{ is continuously differentiable at } \hat{x}, \text{ then } \nabla f(\hat{x}) = 0. \]

... but we do not know anything about \( f \).

We need to work with less restrictive differentiability assumptions.
Coordinate search on a non-differentiable function

\[ f(x) = \|x\|_\infty \text{ with } x_0 = (1, 1)^T. \]

Level set:
\[ \{x \in \mathbb{R}^2 : f(x) = 1\} \]

\[ x_0 = (1, 1)^T \]
Coordinate search on a non-differentiable function

\[ f(x) = \|x\|_\infty \text{ with } x_0 = (1, 1)^T. \]

Level set:
\[ \{ x \in \mathbb{R}^2 : f(x) = 1 \} \]

A directional algorithm
Clarke Calculus

If \( f \) is Lipschitz\(^1 \) near \( \bar{x} \in \mathbb{R}^n \), then Clarke’s generalized derivative at \( \bar{x} \) in the direction \( v \in \mathbb{R}^n \) is

\[
 f^\circ(\bar{x}; v) = \limsup_{y \to \bar{x}, \; t \downarrow 0} \frac{f(y + tv) - f(y)}{t}.
\]

\(^1\)there exists a nonnegative scalar \( K \) such that

\[
|f(x) - f(x')| \leq K \|x - x'\|
\]

for all \( x, x' \) in some open neighborhood of \( \bar{x} \).
Facts on Clarke calculus

♦ The generalized gradient of $f$ at $x$ is the set

$$\partial f(x) := \{ \zeta \in \mathbb{R}^n : f^\circ(x; v) \geq v^T \zeta \text{ for all } v \in \mathbb{R}^n \}.$$ 

♦ Let $f$ be Lipschitz near $x$, then

$$\partial f(x) = \text{co}\{ \lim \nabla f(x_i) : x_i \to x \text{ and } \nabla f(x_i) \text{ exists} \}.$$ 

♦ Generalized derivative can be obtained from the generalized gradient: $f^\circ(x; v) = \max\{ v^T \zeta : \zeta \in \partial f(x) \}$. 

♦ If $x$ is a minimizer of $f$, and $f$ is Lipschitz near $x$, then $0 \in \partial f(x)$. Generalizes the 1st order necessary condition for continuously differentiable $f$: $0 = \nabla f(x)$. 

IMA, Jan 2003
If $f$ is differentiable (Hadamard, Gâteaux, or Fréchet) at $x$, then the derivative of $f$ at $x$ is in the generalized gradient $\partial f(x)$.

When $f$ is convex, $\partial f(x) = \text{subdifferential}$.

$f$ is regular at $x$ if for all $v \in \mathbb{R}^n$, the one-sided directional derivative $f'(x;v)$ exists and equals $f^\circ(x;v)$.

$f$ is strictly differentiable at $x$ if for all $v \in \mathbb{R}^n$,

$$
\lim_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} = \nabla f(x)^T v.
$$

If $f$ is Lipschitz near $x$ and $\partial f(x)$ reduces to a singleton $\{\zeta\}$, then $f$ is strictly differentiable at $x$ and $\nabla f(x) = \zeta$. 

IMA, Jan 2003
Differentiable, not strictly differentiable

\[ y = x^2(2 + \sin(\pi/x)) \]

\( f \) is Lipschitz and differentiable, near 0:

\[ y'(0) = 0 \text{ and } y' = 2x(2 + \sin(\frac{\pi}{x})) - \pi \cos(\frac{\pi}{x}) \]
Differentiable, not strictly differentiable

\[ y = x^2(2 + \sin(\pi/x)) \]

\( f \) is Lipschitz and differentiable, near 0:

\[ y'(0) = 0 \text{ and } y' = 2x(2 + \sin(\frac{\pi}{x})) - \pi \cos(\frac{\pi}{x}) \]

The derivative is not continuous at 0:

\[ y'(\frac{1}{2k}) = \frac{2}{k} - \pi \]
Differentiable, not strictly differentiable

$f$ is Lipschitz and differentiable, near 0:

\[ y'(0) = 0 \text{ and } y' = 2x(2 + \sin(\frac{\pi}{x})) - \pi \cos(\frac{\pi}{x}) \]

The derivative is not continuous at 0:

\[ y'(\frac{1}{2k}) = \frac{2}{k} - \pi \]

$f$ is not strictly differentiable:

\[ \partial f(0) = [-\pi, \pi] \]
Differentiable, not strictly differentiable

\[ y = x^2(2 + \sin(\pi/x)) \]

\( f \) is Lipschitz and differentiable, near 0:
\[ y'(0) = 0 \quad \text{and} \quad y' = 2x(2 + \sin(\frac{\pi}{x})) - \pi \cos(\frac{\pi}{x}) \]

The derivative is not continuous at 0:
\[ y'(\frac{1}{2k}) = \frac{2}{k} - \pi \]

\( f \) is not strictly differentiable:
\[ \partial f(0) = [-\pi, \pi] \]

\( f \) is not regular:
\[ f^\circ(0, \pm 1) = \pi \neq f'(0, \pm 1) = 0 \]

IMA, Jan 2003
Strictly, not continuously differentiable

\[ f(x) = \int_0^x \varphi(u) \, du \quad \text{where} \quad \varphi(u) = \begin{cases} 
  u & \text{if } u \leq 0 \\
  \frac{1}{1+\kappa} & \text{if } \kappa + 1 > \frac{1}{u} \geq \kappa
\end{cases} \]
Strictly, not continuously differentiable

\[ f \text{ is Lipschitz near } \hat{x} = 0, \text{ has kinks at } \frac{1}{\kappa} \text{ with } \partial f \left( \frac{1}{\kappa} \right) = \left[ \frac{1}{\kappa+1}, \frac{1}{\kappa} \right] \]
Strictly, not continuously differentiable

$f$ is Lipschitz near $\hat{x} = 0$, has kinks at $\frac{1}{\kappa}$ with $\partial f(\frac{1}{\kappa}) = [\frac{1}{\kappa + 1}, \frac{1}{\kappa}]$

$f$ is not strictly differentiable, nor continuously differentiable, in any neighborhood of $\hat{x} = 0$. 
Strictly, not continuously differentiable

$f$ is Lipschitz near $\hat{x} = 0$, has kinks at $\frac{1}{\kappa}$ with $\partial f(\frac{1}{\kappa}) = [\frac{1}{\kappa+1}, \frac{1}{\kappa}]$

$f$ is not strictly differentiable, nor continuously differentiable, in any neighborhood of $\hat{x} = 0$.

$\partial f(0) = \{0\}$ therefore $f$ is strictly differentiable at $\hat{x} = 0$. 
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set.

\( \lim_{k} \Delta_k = 0 \)

there is an \( \hat{x} \) which is the limit of a sequence \( \{x_k\}_{k \in K} \) of mesh local optimizers \( (f(x_k \pm \Delta_k e_i) \geq f(x_k) \text{ for all } e_i) \)

If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x}; \pm e_i) \geq 0 \) for every \( e_i \)
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set.

\[ \lim_{k} \Delta_k = 0 \]

\[ \exists \ \text{an } \hat{x} \text{ which is the limit of a sequence } \{x_k\}_{k \in K} \text{ of mesh local optimizers (} f(x_k \pm \Delta_k e_i) \geq f(x_k) \text{ for all } e_i \) \]

\[ \text{If } f \text{ is Lipschitz near } \hat{x}, \text{ then } f^\circ(\hat{x}; \pm e_i) \geq 0 \text{ for every } e_i \]

\[ \text{Proof: } f^\circ(\hat{x}; e_i) := \lim_{y \to \hat{x}, \ t \downarrow 0} \sup \frac{f(y + te_i) - f(y)}{t} \]
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set.

\[ \lim_{k} \Delta_k = 0 \]

there is an \( \hat{x} \) which is the limit of a sequence \( \{x_k\}_{k \in K} \) of mesh local optimizers \( (f(x_k \pm \Delta_k e_i) \geq f(x_k) \text{ for all } e_i) \)

If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x}; \pm e_i) \geq 0 \) for every \( e_i \)

**Proof:**

\[
f^\circ(\hat{x}; e_i) := \limsup_{y \to \hat{x}, \ t \downarrow 0} \frac{f(y + te_i) - f(y)}{t}
\geq \limsup_{k \in K} \frac{f(x_k + \Delta_k e_i) - f(x_k)}{\Delta_k}
\]
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set.

\[ \lim_{k} \Delta_k = 0 \]

there is an \( \hat{x} \) which is the limit of a sequence \( \{x_k\}_{k \in K} \) of mesh local optimizers \( (f(x_k \pm \Delta_k e_i) \geq f(x_k) \text{ for all } e_i) \)

If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x}; \pm e_i) \geq 0 \) for every \( e_i \)

**Proof:**

\[
\begin{align*}
  f^\circ(\hat{x}; e_i) &:= \limsup_{y \to \hat{x}, \, t \downarrow 0} \frac{f(y + te_i) - f(y)}{t} \\
  &\geq \limsup_{k \in K} \frac{f(x_k + \Delta_k e_i) - f(x_k)}{\Delta_k} \\
  &\geq 0
\end{align*}
\]
Convergence of coordinate searches

if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[ \lim_{k} \Delta_k = 0 \]

there is an \( \hat{x} \) which is the limit of a sequence \( \{x_k\}_{k \in K} \) of mesh local optimizers

- If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x}; \pm e_i) \geq 0 \) for every \( e_i \)

- If \( f \) is regular at \( \hat{x} \), then \( f'(\hat{x}; \pm e_i) \geq 0 \) for every \( e_i \)

- If \( f \) is strictly differentiable at \( \hat{x} \), then \( \nabla f(\hat{x}) = 0 \)

IMA, Jan 2003
The two phases of Pattern search algorithms

**Global search** on the mesh

- Flexibility,
- User heuristics,
- Knowledge of the model,
- Surrogate functions.

**Local poll** around the incumbent solution

- More rigidly defined.
- Ensures appropriate first order optimality conditions.
Positive spanning sets

\( D \subset \mathbb{R}^n \) is a positive spanning set if non-negative linear combinations of the elements of \( D \) span \( \mathbb{R}^n \).
Positive spanning sets

$D \subset \mathbb{R}^n$ is a positive spanning set if non-negative linear combinations of the elements of $D$ span $\mathbb{R}^n$. 
Positive spanning sets

$D \subset \mathbb{R}^n$ is a positive spanning set if non-negative linear combinations of the elements of $D$ span $\mathbb{R}^n$. 
**Positive spanning sets**

$D \subset \mathbb{R}^n$ is a positive spanning set if non-negative linear combinations of the elements of $D$ span $\mathbb{R}^n$.

$D$ contains at least $n + 1$ directions
Basic pattern search algorithm for unconstrained optimization

Positive spanning directions: \( D_k = \{d_1, d_2, \ldots, d_p\} \subseteq D \)

Current best point: \( x_k \in \mathbb{R}^n \)

Current mesh size parameter: \( \Delta_k \in \mathbb{R}_+ \)
Basic pattern search algorithm for unconstrained optimization

Positive spanning directions: $D_k = \{d_1, d_2, \ldots, d_p\} \subseteq D$

Current best point: $x_k \in \mathbb{R}^n$

Current mesh size parameter: $\Delta_k \in \mathbb{R}_+$
Search step (global)

Given $\Delta_k, x_k$:

**Search** anywhere on $M_k$. 

$\Delta_k, x_k$: 

$M_k$: 

$x_k$: 

Search step (global)

Given $\Delta_k, x_k$:

**Search** anywhere on $M_k$.

If an **improved mesh point** is found *i.e.* $f(x_{k+1}) < f(x_k)$

then set $\Delta_{k+1} \geq \Delta_k$.
Search step (global)

Given $\Delta_k, x_k$:

**Search** anywhere on $M_k$.

If an **improved mesh point** is found *i.e.* $f(x_{k+1}) < f(x_k)$

then set $\Delta_{k+1} \geq \Delta_k$,

and restart the **search** from this improved point.
Poll step (local)

If \texttt{SEARCH} fails, \texttt{POLL} at mesh neighbors:
Poll step (local)

If search fails,
**Poll** at mesh neighbors:

If an **improved mesh point**
is found *i.e.* $f(x_{k+1}) < f(x_k)$,
set $\Delta_{k+1} \geq \Delta_k$, and restart
search from improved point.
Poll step (local)

If \textsc{search} fails, \textbf{Poll} at mesh neighbors:

If an \textsc{improved mesh point} is found \textit{i.e.} \( f(x_{k+1}) < f(x_k) \), set \( \Delta_{k+1} \geq \Delta_k \), and restart \textsc{search} from improved point.

Else \( x_{k+1} = x_k \) is a \textsc{mesh local optimizer}. Set \( \Delta_{k+1} < \Delta_k \), and restart \textsc{search} from this point.
Convergence results

If all iterates are in a compact set then

\[ \liminf_k \Delta_k = 0 \] (the proof is non trivial since \( \Delta_{k+1} \) may increase)
Convergence results

If all iterates are in a compact set then

- $\liminf_{k} \Delta_k = 0$ (the proof is non trivial since $\Delta_{k+1}$ may increase)
- For every limit point $\hat{x}$ of any subsequence $\{x_k\}_{k \in K}$ of mesh local optimizers where $\{\Delta_k\}_{k \in K} \to 0$, and for the set $\hat{D}$ of POLL directions used infinitely many times in this subsequence
Convergence results

If all iterates are in a compact set then

- \( \liminf_k \Delta_k = 0 \) (the proof is non trivial since \( \Delta_{k+1} \) may increase)
- For every limit point \( \hat{x} \) of any subsequence \( \{x_k\}_{k \in K} \) of mesh local optimizers where \( \{\Delta_k\}_{k \in K} \to 0 \), and for the set \( \hat{D} \) of POLL directions used infinitely many times in this subsequence

If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x};d) \geq 0 \) for every \( d \in \hat{D} \)

If \( f \) is regular at \( \hat{x} \), then \( f'(\hat{x};d) \geq 0 \) for every \( d \in \hat{D} \).

If \( f \) is strictly differentiable at \( \hat{x} \), then \( \nabla f(\hat{x}) = 0 \).


The convergence results do not state that if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[ \lim_{k} \Delta_k = 0 \]
The convergence results do **not** state that if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[ \lim_{k} \Delta_k = 0 \]

If \( f \) is continuously differentiable everywhere then \( \nabla f(\hat{x}) = 0 \) for any limit point \( \hat{x} \) of the sequence of iterates.
The convergence results do not state that if the sequence of iterates \( \{x_k\} \) belongs to a compact set

\[ \lim_{k} \Delta_k = 0 \]

- If \( f \) is continuously differentiable everywhere then \( \nabla f(\hat{x}) = 0 \) for any limit point \( \hat{x} \) of the sequence of iterates.

- If the entire sequence of iterates converges (at \( \hat{x} \) say), and if \( f \) is differentiable then \( \nabla f(\hat{x}) = 0 \).
The convergence results do not state that
if the sequence of iterates \( \{x_k\} \) belongs to a compact set
\( \lim_{k} \Delta_k = 0 \)

If \( f \) is continuously differentiable everywhere then
\( \nabla f(\hat{x}) = 0 \) for any limit point \( \hat{x} \) of the sequence of iterates.

If the entire sequence of iterates converges (at \( \hat{x} \) say),
and if \( f \) is differentiable then \( \nabla f(\hat{x}) = 0 \).

If the entire sequence of iterates converges (at \( \hat{x} \) say), and if \( f \)
is Lipschitz near \( \hat{x} \) then \( 0 \in \partial f(\hat{x}) \). Thus, the method does not
necessarily produce a stationary point in the Clarke sense.
Basic pattern search algorithm for linearly constrained optimization

♦ Infeasible trial points are pruned (the objective function is not evaluated and set to infinity).

♦ When $x_k$ is close to the boundary of the feasible region, the POLL directions must conform to the boundary.
Convergence results – linearly constraints

If all iterates are in a compact set then

\[ \liminf_k \Delta_k = 0 \]

For every limit point \( \hat{x} \) of any subsequence \( \{x_k\}_{k \in K} \) of mesh local optimizers where \( \{\Delta_k\}_{k \in K} \to 0 \), and for the set \( \hat{D} \) of POLL directions used infinitely many times in this subsequence
Convergence results – linearly constraints

If all iterates are in a compact set then

\[ \liminf_k \Delta_k = 0 \]

For every limit point \( \hat{x} \) of any subsequence \( \{x_k\}_{k \in K} \) of mesh local optimizers where \( \{\Delta_k\}_{k \in K} \to 0 \), and for the set \( \hat{D} \) of POLL directions used infinitely many times in this subsequence

- If \( f \) is Lipschitz near \( \hat{x} \), then \( f^\circ(\hat{x}; d) \geq 0 \) for every \( d \in \hat{D} \cap TX(\hat{x}) \).

- If \( f \) is regular at \( \hat{x} \), then \( f'(\hat{x}; d) \geq 0 \) for every \( d \in \hat{D} \cap TX(\hat{x}) \).

- If \( f \) is s.d. at \( \hat{x} \), then \( \nabla f(\hat{x})^T v \geq 0 \) for every \( v \in TX(\hat{x}) \).
Positive spanning sets

♦ A richer set of $D$ increases the number of directions in $\hat{D}$, and thus the directions for which $f^\circ(\hat{x};d) \geq 0$.

♦ User’s domain specific knowledge can help choose $D_k \subset D$.

♦ Theory limited to a finite number of directions in $D$, so the barrier approach (reject infeasible points) does not apply to general constraints because a finite number can not conform to the boundary of $\Omega$. 
General nonlinear constrained optimization

\[
\begin{align*}
\min_{x \in X} & \quad f(x) \\
\text{s.t.} & \quad x \in \Omega \equiv \{ x | C(x) \leq 0 \}, \quad \text{where } C = (c_1, c_2, \ldots, c_m)^T
\end{align*}
\]

$X$ is defined by bound and linear constraints.
Define the nonnegative constraint violation function $h : \mathbb{R}^n \to \mathbb{R}$

\[
h(x) = \sum_{j} \max(0, c_j(x))^2.
\]

Note that $h(x) = 0$ iff $x \in \Omega$, and $h$ inherits smoothness from $C$. 
General nonlinear constrained optimization

\[
\min_{x \in X} \ f(x)
\]
\[
\text{s.t. } x \in \Omega \equiv \{ x \mid C(x) \leq 0 \}, \quad \text{where } C = (c_1, c_2, \ldots, c_m)^T
\]

\(X\) is defined by bound and linear constraints.

Define the nonnegative constraint violation function \(h : \mathbb{R}^n \rightarrow \mathbb{R}\)

\[
h(x) = \sum_j \max(0, c_j(x))^2.
\]

Note that \(h(x) = 0\) iff \(x \in \Omega\), and \(h\) inherits smoothness from \(C\).

We look at the biobjective optimization problem where a priority is given to the minimization of \(h\) over the minimization of \(f\).
Pattern Search with filter
Pattern Search with filter

\[ f = 28 \]
Pattern Search with filter

\[ f = 28 \]
Pattern Search with filter

\( f = 28 \)
Pattern Search with filter

\begin{align*}
\text{\( f = 28 \)} & \quad \text{\( f = 34 \)}
\end{align*}
Pattern Search with filter

\[ f = 28 \quad f = 34 \]
Pattern Search with filter

- $f = 28$
- $f = 34$
- $f = 38$
Pattern Search with filter

\[ f = 28 \]
\[ f = 34 \]
\[ f = 38 \]
Pattern Search with filter

- \( f = 28 \)
- \( f = 24 \)
- \( f = 34 \)
- \( f = 38 \)

Graph showing points on a grid with values for \( f \) and \( h \).
Pattern Search with filter

- $f = 24$
- $f = 28$
- $f = 34$
- $f = 38$

$h$
Pattern Search with filter

\[ f = 24 \]
\[ f = 28 \]
\[ f = 34 \]
\[ f = 38 \]
Pattern Search with filter

\[ f = 38 \]
\[ f = 24 \]
\[ f = 28 \]
\[ f = 34 \]
\[ f = 15 \]
Pattern Search with filter

\( f = 15 \)

\( f = 24 \)

\( f = 28 \)

\( f = 34 \)

\( f = 38 \)
Pattern Search with filter

\[ f = 24 \]
\[ f = 28 \]
\[ f = 34 \]
\[ f = 38 \]
\[ f = 40 \]

\[ f = 15 \]
Pattern Search with filter

\[ f = 20 \]
\[ f = 15 \]
\[ f = 24 \]
\[ f = 28 \]
\[ f = 34 \]
\[ f = 38 \]
Pattern Search with filter

\[ f = 15 \]

\[ f = 20 \]

\[ f = 24 \]

\[ f = 28 \]

\[ f = 34 \]

\[ f = 38 \]
Pattern Search with filter

\begin{itemize}
  \item $f = 24$
  \item $f = 28$
  \item $f = 34$
  \item $f = 20$
  \item $f = 15$
  \item $f = 18$
\end{itemize}
Pattern Search with filter

\[ f = 20 \quad f = 15 \quad f = 18 \]

\[ f = 24 \quad f = 28 \quad f = 34 \]

\[ f = 28 \quad f = 34 \quad f = 38 \]

\[ f = 15 \quad f = 20 \quad f = 18 \]
Pattern Search with filter

\[ f = 24 \]

\[ f = 15 \]

\[ f = 18 \]
Pattern Search with Filter

\[ f = 24 \]

\[ f = 18 \]
Pattern Search with filter

\[ f = 24 \]

\[ f = 18 \]

\[ f = 16 \]
Pattern Search with filter

\[ f = 24 \]
\[ f = 18 \]
\[ f = 16 \]
\[ h = \infty \]
Pattern Search with filter

- $f = 24$
- $f = 17$
- $f = 18$
- $f = 16$
- $h = \infty$

![Graph showing the pattern search with filter](image)
Pattern Search with filter
Pattern Search with filter

\[ f = 24 \]

\[ f = 17 \]
Pattern Search with filter

\[ f = 24 \]

\[ f = 25 \]

\[ f = 17 \]
Pattern Search with filter

\[ f = 24 \]
\[ f = 20 \]
\[ f = 25 \]
\[ f = 17 \]
Pattern Search with filter

\[ f = 24 \]
\[ f = 20 \]
\[ f = 25 \]
\[ f = 17 \]
\[ f = 18 \]
Pattern Search with filter

Mesh isolated filter point
Convergence results – general constraints

Polling around least infeasible point gives priority to the minimization of $h$ versus the minimization of $f$.

If all iterates are in a compact set then

\[ \liminf_k \Delta_k = 0 \]

For every limit point $\hat{p}$ of any subsequence \( \{p_k\}_{k \in K} \) of mesh isolated poll centers where \( \{\Delta_k\}_{k \in K} \to 0 \), and for the set $\hat{D}$ of POLL directions used infinitely many times in this subsequence
Convergence results – general constraints

Polling around least infeasible point gives priority to the minimization of \( h \) versus the minimization of \( f \).

If all iterates are in a compact set then

\[ \liminf_{k} \Delta_k = 0 \]

For every limit point \( \hat{p} \) of any subsequence \( \{p_k\}_{k \in K} \) of mesh isolated poll centers where \( \{\Delta_k\}_{k \in K} \to 0 \), and for the set \( \hat{D} \) of POLL directions used infinitely many times in this subsequence

\[ h \circ (\hat{x}; v) \geq 0 \text{ for every } v \in T_X(\hat{p}) \]
Convergence results – general constraints

Polling around least infeasible point gives priority to the minimization of $h$ versus the minimization of $f$.

If all iterates are in a compact set then

- $\liminf_{k} \Delta_k = 0$
- For every limit point $\hat{p}$ of any subsequence $\{p_k\}_{k \in K}$ of mesh isolated poll centers where $\{\Delta_k\}_{k \in K} \to 0$, and for the set $\hat{D}$ of poll directions used infinitely many times in this subsequence

If $h$ is Lipschitz near $\hat{p}$, and $\hat{p}$ is feasible then $h^\circ(\hat{x}; v) \geq 0$ for every $v \in T_X(\hat{p})$

If $h$ is Lipschitz near $\hat{p}$, then $h^\circ(\hat{p}; d) \geq 0$ for every $d \in \hat{D} \cap T_X(\hat{p})$
If \( h \) is regular at \( \hat{p} \), then \( h'(\hat{x}; d) \geq 0 \) for every \( d \in \hat{D} \cap TX(\hat{p}) \).

If \( h \) is s.d. at \( \hat{p} \), then \( \nabla h(\hat{x})^Tv \geq 0 \) for every \( v \in TX(\hat{x}) \).

Note: The same convergence results hold for \( f \), with an additional requirement: \( \hat{p} \) must be strictly feasible with respect to the general constraints.
A planform filter: 17 vars, 13 ctrs, kriging
A planform filter: 17 vars, 13 ctrs, kriging
A planform filter: 17 vars, 13 ctrs, kriging
Discussion

♦ GPS algorithms are directional algorithms:
   Analysis is easy with Clarke calculus
Discussion

♦ GPS algorithms are directional algorithms: Analysis is easy with Clarke calculus

♦ Optimality results depend on local differentiability The analysis can be extended by considering other directions
Discussion

- GPS algorithms are directional algorithms:
  Analysis is easy with Clarke calculus

- Optimality results depend on local differentiability
  The analysis can be extended by considering other directions

- Linear and bound constraints are treated by appropriate polling directions
Discussion

♦ GPS algorithms are directional algorithms:
   Analysis is easy with Clarke calculus

♦ Optimality results depend on local differentiability
   The analysis can be extended by considering other directions

♦ Linear and bound constraints are treated by appropriate polling directions

♦ General constraints are treated by the filter
Discussion

♦ GPS algorithms are directional algorithms: Analysis is easy with Clarke calculus

♦ Optimality results depend on local differentiability The analysis can be extended by considering other directions

♦ Linear and bound constraints are treated by appropriate polling directions

♦ General constraints are treated by the filter

♦ GPS for problems with categorical variables

Tomorrow’s talk by Mark Abramsom