A trust region method for solving grey-box mixed integer nonlinear problems with industrial applications.

Andrew R. Conn
arconn@us.ibm.com

IBM T.J. Watson Research Center

joint work with Claudia D'Ambrosio, Leo Liberti, and Delphine Sinoquet.

23-25 mars, 2016,

Toulouse, France

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Reality

- One almost never sees asymptotics
- One almost never reaches a solution but even 1% improvement can be extremely valuable
- Because of their complexity, simulation is often required
- Because of simulation derivatives are often not available
- It is always better to obtain and use derivatives if you can.
- Simulated Annealing, Genetic Algorithms etc are usually for the ignorant or the desperate or both.

Some remarks:

1. the function can have many local minima,
2. the value of the function can include both noise and error
3. the evaluation of the function can be expensive,
4. the domain of the function can be unknown.

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.

  Second Order

$$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
$$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
$$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$

- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).
- By reducing the trust region or step size one guarantees better accuracy.

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.

  First Order

$$|f(x) - m(x)| \leq \mathcal{O}(\Delta^2)$$
$$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta)$$

  Second Order

$$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
$$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
$$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$

- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.
  Second Order
  $$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
  $$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
  $$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$
- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).
- By reducing the trust region or step size one guarantees better accuracy.

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.

  Second Order

$$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
$$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
$$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$

- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).
- By reducing the trust region or step size one guarantees better accuracy.

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.
  Second Order
  $$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
  $$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
  $$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$
- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).
- By reducing the trust region or step size one guarantees better accuracy.

Standard Model in the differentiable case

- Typical trust region or line search method builds linear or quadratic model of the objective function f.
- The model has to satisfy Taylor-like error bounds.
  Second Order
  $$|f(x) - m(x)| \leq \mathcal{O}(\Delta^3)$$
  $$|\nabla f(x) - \nabla m(x)| \leq \mathcal{O}(\Delta^2)$$
  $$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \mathcal{O}(\Delta)$$
- In fact it typically is a first (or second) order Taylor series approximation.
- In derivative based methods constants in $\mathcal{O}$ depend only on f (and its derivatives).
- By reducing the trust region or step size one guarantees better accuracy.

Abstraction of required bounds for derivative free

- $f \in C^1$ and ∇f Lipschitz continuous on $\{x | f_k \leq f_0\}$.
- Δ_k bounded above.
- A model is called:
  Fully Linear on $B(x, \Delta)$ iff

$$|f(x) - m(x)| \leq \kappa_{ef} \Delta^2$$

$$|\nabla f(x) - \nabla m(x)| \leq \kappa_{eg} \Delta$$

  for all x in $B(x, \Delta)$
- A Fully Linear model that is suitably minimized replaces the Cauchy Point.

Abstraction of required bounds for derivative free

- $f \in C^1$ and ∇f Lipschitz continuous on $\{x | f_k \leq f_0\}$.
- Δ_k bounded above.
- A model is called:
  Fully Linear on $B(x, \Delta)$ iff

  $$|f(x) - m(x)| \leq \kappa_{ef} \Delta^2$$

  $$|\nabla f(x) - \nabla m(x)| \leq \kappa_{eg} \Delta$$

  for all x in $B(x, \Delta)$
- A Fully Linear model that is suitably minimized replaces the Cauchy Point.

Abstraction of required bounds

- $f \in C^2$ and $\nabla^2 f$ Lipschitz continuous on $\{x | f_k \leq f_0\}$.
- Δ_k bounded above.
- A model is called:
  Fully Quadratic on $B(x, \Delta)$ iff

$$|f(x) - m(x)| \leq \kappa_{ef} \Delta^3$$

$$|\nabla f(x) - \nabla m(x)| \leq \kappa_{eg} \Delta^2$$

$$|\nabla^2 f(x) - \nabla^2 m(x)| \leq \kappa_{eh} \Delta$$

for all x in $B(x, \Delta)$

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

So what about the case without derivatives?

- Model depends on previous iterates!
- **Geometry matters**

- In derivative free methods we use sample based models; e.g., interpolation or regression or pattern-based methods.
- The $\mathcal{O}$ in Taylor-like bounds depends not only on f, but also on the geometry of the sample set.
- We need to have some constant characterizing the quality of the sample set(automatic in pattern-based methods) .
- We need to control this constant to keep it uniformly bounded.

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\quad \Delta \leftarrow 2.0\Delta$ $\quad$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\quad \Delta \leftarrow \Delta$ $\quad$ Accept $x_k + s_k$

If $\rho < 0.25$ $\quad \Delta \leftarrow 0.5\Delta$ $\quad$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\quad \Delta \leftarrow 2.0\Delta$ $\quad$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\quad \Delta \leftarrow \Delta$ $\quad$ Accept $x_k + s_k$

If $\rho < 0.25$ $\quad \Delta \leftarrow 0.5\Delta$ $\quad$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\Delta \leftarrow 2.0\Delta$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\Delta \leftarrow \Delta$ Accept $x_k + s_k$

If $\rho < 0.25$ $\Delta \leftarrow 0.5\Delta$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\quad \Delta \leftarrow 2.0\Delta$ $\quad$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\quad \Delta \leftarrow \Delta$ $\quad$ Accept $x_k + s_k$

If $\rho < 0.25$ $\quad \Delta \leftarrow 0.5\Delta$ $\quad$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\quad \Delta \leftarrow 2.0\Delta$ $\quad$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\quad \Delta \leftarrow \Delta$ $\quad$ Accept $x_k + s_k$

If $\rho < 0.25$ $\quad \Delta \leftarrow 0.5\Delta$ $\quad$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Basic Algorithm when derivatives available (unconstrained)

Initialize: x_0, Δ

Compute Model: $m_k(\ )$

Compute Step: Compute s_k from

$$\min_{\|s\| \leq \Delta} m_k(x_k + s)$$

Trust-region Update: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$

If $\rho > 0.75$ $\quad \Delta \leftarrow 2.0\Delta$ $\quad$ Accept $x_k + s_k$

If $0.25 < \rho < 0.75$ $\quad \Delta \leftarrow \Delta$ $\quad$ Accept $x_k + s_k$

If $\rho < 0.25$ $\quad \Delta \leftarrow 0.5\Delta$ $\quad$ Reject $x_k + s_k$

NOTE: We always reduce TR when we reject

Model decrease at the Cauchy point

Fundamental result that drives convergence:

$$m_k(x_k) - m_k(x_k^C) \geq \frac{1}{2}\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right],$$

where

$$g_k = \nabla_x m_k(x_k), \quad \beta_k = 1 + \max_{x \in B_k} \|\nabla_{xx} m_k(x)\|$$

$$x_k^C(t) = \{x \mid x = x_k - t g_k, \; t \geq 0 \quad \text{and} \quad x \in B_k\} \quad \text{and} \quad x_k^C = \text{argmin}\, m_k(x_k^C(t)).$$

$\Rightarrow$ Define

sufficient model decrease

$$\Leftrightarrow$$

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa \|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right]$$

Model decrease at the Cauchy point

Fundamental result that drives convergence:

$$m_k(x_k) - m_k(x_k^{\mathrm{C}}) \geq \frac{1}{2}\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right],$$

where

$$g_k = \nabla_x m_k(x_k), \quad \beta_k = 1 + \max_{x \in B_k} \|\nabla_{xx} m_k(x)\|$$

$$x_k^{\mathrm{C}}(t) = \{x \mid x = x_k - t g_k, \ t \geq 0 \quad \text{and} \quad x \in B_k\} \quad \text{and} \quad x_k^{\mathrm{C}} = \operatorname{argmin} m_k(x_k^{\mathrm{C}}(t)).$$

$\Rightarrow$ Define

sufficient model decrease

$$\Leftrightarrow$$

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa \|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right]$$

Model decrease at the Cauchy point

Fundamental result that drives convergence:

$$m_k(x_k) - m_k(x_k^C) \geq \frac{1}{2}\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right],$$

where

$$g_k = \nabla_x m_k(x_k), \quad \beta_k = 1 + \max_{x \in B_k} \|\nabla_{xx} m_k(x)\|$$

$$x_k^C(t) = \{x \mid x = x_k - tg_k, \ t \geq 0 \quad \text{and} \quad x \in B_k\} \quad \text{and} \quad x_k^C = \text{argmin}\, m_k(x_k^C(t)).$$

$\Rightarrow$ Define

sufficient model decrease

$\Leftrightarrow$

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right]$$

Model decrease at the Cauchy point

Fundamental result that drives convergence:

$$m_k(x_k) - m_k(x_k^C) \geq \frac{1}{2}\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right],$$

where

$$g_k = \nabla_x m_k(x_k), \quad \beta_k = 1 + \max_{x \in B_k} \|\nabla_{xx} m_k(x)\|$$

$$x_k^C(t) = \{x \mid x = x_k - tg_k, \; t \geq 0 \quad \text{and} \quad x \in B_k\} \quad \text{and} \quad x_k^C = \text{argmin}\, m_k(x_k^C(t)).$$

$\Rightarrow$ Define

sufficient model decrease

$\Leftrightarrow$

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa\|g_k\| \min\left[\frac{\|g_k\|}{\beta_k}, \Delta_k\right]$$

Trust-region Methods without derivatives

- Use interpolation/regression models that mimic the Taylor series expansions.
- We never reduce the trust region radius Δ_k if the sample set is badly-poised (has bad geometry).
- Incorporate a stationarity condition (first or second order) when 'stationarity' of the model is sufficiently small.

  $\longrightarrow$ Iterative process with successive contractions of Δ_k.
- Converged when the radius is small enough.
- The problem only has to be reasonably approximated by a sufficiently smooth problem.

Trust-region Methods without derivatives

- Use interpolation/regression models that mimic the Taylor series expansions.
- We never reduce the trust region radius Δ_k if the sample set is badly-poised (has bad geometry).
- Incorporate a stationarity condition (first or second order) when 'stationarity' of the model is sufficiently small.

  $\longrightarrow$ Iterative process with successive contractions of Δ_k.
- Converged when the radius is small enough.
- The problem only has to be reasonably approximated by a sufficiently smooth problem.

Trust-region Methods without derivatives

- Use interpolation/regression models that mimic the Taylor series expansions.

- We never reduce the trust region radius Δ_k if the sample set is badly-poised (has bad geometry).

- Incorporate a stationarity condition (first or second order) when 'stationarity' of the model is sufficiently small.

  $\longrightarrow$ Iterative process with successive contractions of Δ_k.

- Converged when the radius is small enough.

- The problem only has to be reasonably approximated by a sufficiently smooth problem.

Trust-region Methods without derivatives

- Use interpolation/regression models that mimic the Taylor series expansions.
- We never reduce the trust region radius Δ_k if the sample set is badly-poised (has bad geometry).
- Incorporate a stationarity condition (first or second order) when 'stationarity' of the model is sufficiently small.

  $\longrightarrow$ Iterative process with successive contractions of Δ_k.
- Converged when the radius is small enough.
- The problem only has to be reasonably approximated by a sufficiently smooth problem.

Trust-region Methods without derivatives

- Use interpolation/regression models that mimic the Taylor series expansions.

- We never reduce the trust region radius Δ_k if the sample set is badly-poised (has bad geometry).

- Incorporate a stationarity condition (first or second order) when 'stationarity' of the model is sufficiently small.

  $\longrightarrow$ Iterative process with successive contractions of Δ_k.

- Converged when the radius is small enough.

- The problem only has to be reasonably approximated by a sufficiently smooth problem.

Grey-box MINLP

$$\min_{x,y} \; S(x,y) + f(x,y)$$

subject to

$$\begin{aligned} \phi(x,y) &\leq 0 \\ Ax + By &\leq b \\ x &\in [x^L, x^U] \\ y &\in \{0,1\}^q, \end{aligned}$$

- $x \in \mathbb{R}^p, y \in \{0,1\}^q$ are decision variables
- $S : \mathbb{R}^n \rightarrow \mathbb{R}$ is a black-box function.
- $f : \mathbb{R}^{p+q} \rightarrow \mathbb{R}$ and $\phi : \mathbb{R}^{p+q} \rightarrow \mathbb{R}^r$ are closed-form functions.
- assumption: S (for relaxed y), f, and ϕ are twice differentiable;

Define $F(x,y) = S(x,y) + f(x,y)$

Grey-box MINLP

$$\min_{x,y} \; S(x,y) + f(x,y)$$

subject to

$$\begin{aligned} \phi(x,y) &\leq 0 \\ Ax + By &\leq b \\ x &\in [x^L, x^U] \\ y &\in \{0,1\}^q, \end{aligned}$$

- $x \in \mathbb{R}^p, y \in \{0,1\}^q$ are decision variables
- $S : \mathbb{R}^n \rightarrow \mathbb{R}$ is a black-box function.
- $f : \mathbb{R}^{p+q} \rightarrow \mathbb{R}$ and $\phi : \mathbb{R}^{p+q} \rightarrow \mathbb{R}^r$ are closed-form functions.
- assumption: S (for relaxed y), f, and ϕ are twice differentiable;

Define $F(x,y) = S(x,y) + f(x,y)$

Grey-box MINLP

$$\min_{x,y} \; S(x,y) + f(x,y)$$

subject to

$$\begin{aligned} \phi(x,y) &\leq \; 0 \\ Ax + By &\leq \; b \\ x &\in \; [x^L, x^U] \\ y &\in \; \{0,1\}^q, \end{aligned}$$

- $x \in \mathbb{R}^p, y \in \{0,1\}^q$ are decision variables
- $S : \mathbb{R}^n \to \mathbb{R}$ is a black-box function.
- $f : \mathbb{R}^{p+q} \to \mathbb{R}$ and $\phi : \mathbb{R}^{p+q} \to \mathbb{R}^r$ are closed-form functions.
- assumption: S (for relaxed y), f, and ϕ are twice differentiable;

Define $F(x,y) = S(x,y) + f(x,y)$

Trust region methods for grey-box MINLP

Subproblem trust region and constraint

1. center (x^k, y^k): select from the previous iterate.
2. trust region for x: a ball centered at x^k (normally) in l_∞, i.e TR is a box $[\underline{x}, \overline{x}]$.
3. trust region subproblem constraint for y: local branching constraint to limit the number of flips in binary variables

$$\sum_{\{j:\ y_j^k=0\}} y_j + \sum_{\{j:\ y_j^k=1\}} (1 - y_j) \leq k.$$

How to avoid redundant space exploration: no-good cuts

Suppose the following situation

1. We trust our model in the current region
2. We have a current best point (x', y')
3. We cannot improve the current point.

Need to look for other local minima

1. Additional trust region subproblem constraint for y: local branching constraint to act as a cut

$$\sum_{\{j:\ y_j^k=0\}} y_j + \sum_{\{j:\ y_j^k=1\}} (1 - y_j) \geq k + 1.$$

2. We will have a bunch of cuts, one for each "sufficiently explored" region.
3. As soon as a new current point is found, we can restore the local branching constraint.
4. We use no-good cuts to mimic the pruning process of branch-and-bound.

Trust region methods for grey-box MINLP

Model for f

1. Taylor series approximation.
2. For example

$$f_{\mathcal{M}}(x,y) = a + b^T x + c^T y + \frac{1}{2}(x,y)^T \,\Omega\, (x,y).$$

Model for S

1. Linear or quadratic function
2. $$S_{\mathcal{M}}(x,y) = \alpha + \beta^T x + \gamma^T y + \frac{1}{2}(x,y)^T \,\Gamma\, (x,y).$$
3. Found by regression or interpolation.

Trust region methods for grey-box MINLP

Putting it all together:the overall trust region subproblem

$$\min_{x,y} \; S_{\mathcal{M}}(x,y) + f_{\mathcal{M}}(x,y)$$

subject to

$$\begin{aligned}
\phi(x,y) \leq & \quad 0 \\
Ax + By \leq & \quad b \\
x \in & \quad [x^L, x^U] \cap [\underline{x}, \overline{x}] \\
y \in & \quad \{0,1\}^q \\
\sum_{\{j:\, y_j^*=0\}} y_j + \sum_{\{j:\, y_j^*=1\}} (1-y_j) \leq & \quad k.
\end{aligned}$$

Global Convergence

For simplicity of explanation I will assume that the constraints

$$\phi(x, y) \leq 0$$
$$Ax + By \leq b$$

are absent.

We first need to define a modified version of the Cauchy step.

Since we have a mixture of discrete and continuous variables we consider such a direction for fixed discrete variables.

Thus, we define the modified Cauchy step $s_k^{y,C}$, **for fixed** y

$$t_k^{y,C} = \operatorname{argmin}_{t \geq 0 : x_k - tg_k \in B(x_k; y; \Delta_k) \cap [x^L, x^U]} m_k(x_k - tg_k, y),$$

$B(x_k; y; \Delta_k)$ is the TR, y is fixed, $m_k(x_k, y)$ is the current model for $f_{\mathscr{M}}(x_k, y) + S_{\mathscr{M}}(x_k, y)$ and $g_k = \nabla_x m_k(x_k, y)$ is the gradient wrt x.

Global Convergence

For simplicity of explanation I will assume that the constraints

$$\phi(x,y) \leq 0$$
$$Ax + By \leq b$$

are absent.

We first need to define a modified version of the Cauchy step.

Since we have a mixture of discrete and continuous variables we consider such a direction for fixed discrete variables.

Thus, we define the modified Cauchy step $s_k^{y,C}$, **for fixed** y

$$t_k^{y,C} = \operatorname{argmin}_{t \geq 0 : x_k - tg_k \in B(x_k; y; \Delta_k) \cap [x^L, x^U]} m_k(x_k - tg_k, y),$$

$B(x_k; y; \Delta_k)$ is the TR, y is fixed, $m_k(x_k, y)$ is the current model for $f_{\mathscr{M}}(x_k, y) + S_{\mathscr{M}}(x_k, y)$ and $g_k = \nabla_x m_k(x_k, y)$ is the gradient wrt x.

Global Convergence

For simplicity of explanation I will assume that the constraints

$$\phi(x, y) \leq 0$$
$$Ax + By \leq b$$

are absent.

We first need to define a modified version of the Cauchy step.

Since we have a mixture of discrete and continuous variables we consider such a direction for fixed discrete variables.

Thus, we define the modified Cauchy step $s_k^{y,C}$, **for fixed** y

$$t_k^{y,C} = \text{argmin}_{t \geq 0: x_k - tg_k \in B(x_k; y; \Delta_k) \cap [x^L, x^U]} m_k(x_k - tg_k, y),$$

$B(x_k; y; \Delta_k)$ is the TR, y is fixed, $m_k(x_k, y)$ is the current model for $f_{\mathscr{M}}(x_k, y) + S_{\mathscr{M}}(x_k, y)$ and $g_k = \nabla_x m_k(x_k, y)$ is the gradient wrt x.

Global Convergence

For simplicity of explanation I will assume that the constraints

$$\phi(x, y) \leq 0$$
$$Ax + By \leq b$$

are absent.

We first need to define a modified version of the Cauchy step.
Since we have a mixture of discrete and continuous variables we consider such a direction for fixed discrete variables.
Thus, we define the modified Cauchy step $s_k^{y,C}$, **for fixed** y

$$t_k^{y,C} = \text{argmin}_{t \geq 0 : x_k - tg_k \in B(x_k;y;\Delta_k) \cap [x^L, x^U]} \, m_k(x_k - tg_k, y),$$

$B(x_k; y; \Delta_k)$ is the TR, y is fixed, $m_k(x_k, y)$ is the current model for $f_{\mathcal{M}}(x_k, y) + S_{\mathcal{M}}(x_k, y)$ and $g_k = \nabla_x m_k(x_k, y)$ is the gradient wrt x.

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

Global Convergence: Special Issues

We define our fully linear or fully quadratic models in x and y, as if the y are relaxed.

The Algorithm and theory are developed for fixed y.

But when we solve the trust region subproblem for y not fixed we solve it as a mixed integer problem.

So eventually we have the correct y and the correct (local) solution.

But note that we may sometimes need an additional evaluation of truth and this could take at least n function evaluations

Generally constraints are included in the subproblem and Cauchy point definition

The Algorithm -1^{st} order version

Step 0: Initialization. Choose a FL class of models and a corresponding MIA. Choose x_0, y_0 (feasible) and Δ_{max}, $\Delta_0^{icb} \in (0, \Delta_{\max})$, and a initial model m_0^{icb}. and the constants η_0, η_1, γ, γ_{inc}, ϵ_c, β, μ, and ω with $0 \leq \eta_0 \leq \eta_1 < 1$ (with $\eta_1 \neq 0$), $0 < \gamma < 1 < \gamma_{inc}$, $\epsilon_c > 0$, $\mu > \beta > 0$, and $\omega \in (0, 1)$. Set $k = 0$.

Step 1: Criticality test. If $\|g_k^{m,inc}\| > \epsilon_c$ then $m_k = m_k^{icb}$ and $\Delta_k = \Delta_k^{icb}$. Otherwise call the MIA to certify m_k^{icb} is FL on $B(x_k; y; \Delta_k^{icb})$.

If $g_k^{m,inc} \leq \epsilon_c$ and at least one of

- the model m_k^{icb} is not certifiably FL on $B(x_k; y; \Delta_k^{icb})$,
- $\Delta_k^{icb} > \mu \|g_k^{m,inc}\|$,

holds then apply a criticality step algorithm to construct a model that is FL on a suitably small region, the ball $B(x_k; y; \tilde{\Delta}_k)$, for some $\tilde{\Delta}_k \in (0, \mu\|\tilde{g}_k\|]$

Otherwise set $m_k = m_k^{icb}$ and $\Delta_k = \Delta_k^{icb}$.

Step 2: Step calculation. Choose a step s_k that (sufficiently) reduces the model $m_k(x, y)$ (approximate CP) such that $x_k + s_k \in B_k(x_k; y; \Delta_k)$.

The Algorithm (continued)

Step 3: Acceptance of the trial point. Compute $F(x_k + s_k, y)$ and $\rho_k = \frac{F(x_k,y) - F(x_k+s_k,y)}{m_k(x_k,y) - m_k(x_k+s_k,y)}$.
If $\rho_k > \eta_1$ or $\rho_k > \eta_0$ and m_k is FL on $B(x_k; y; \Delta_k)$, then $x_{k+1} = x_k + s_k$, and the model is updated; otherwise the model and the iterate remain unchanged.

Step 4: Model improvement. If $\rho_k < \eta_1$ use MIA to attempt to certify that m_k is FL on $B(x_k; y; \Delta_k)$.
If such a certificate is not obtained, we say that m_k is not certifiably FL and make suitable improvement steps.

Define m_{k+1}^{icb} to be the (possibly improved) model.

The Algorithm (continued)

Step 5: Possible second step calculation. As long as $x_{k+1} \neq x_k$.
Choose a step $\tilde{s}_k$ that (sufficiently) reduces the model $m_k(x_{k+1}, y_k)$ such that $(x_{k+1}, y_k) + \tilde{s}_k \in B_k(x_k; y_k; \Delta_k)$. Note y **is not fixed**
Set $(x_{k+1}, y_{k+1}) = (x_{k+1}, y_k) + \tilde{s}_k$

Step 6: Trust-region radius update. Set

$$\Delta_{k+1}^{icb} \in \begin{cases} [\Delta_k, \min\{\gamma_{inc}\Delta_k, \Delta_{max}\}] & \text{if } \rho_k \geq \eta_1, \\ \{\gamma\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ is FL}, \\ \{\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \\ & \text{is not certifiably fully linear.} \end{cases}$$

Increment k by one and go to Step 1.

Note: the model-improvement algorithm to improve the model until it is fully linear on the required trust region can be done in a finite, uniformly bounded number of steps .

The Algorithm (continued)

Step 5: Possible second step calculation. As long as $x_{k+1} \neq x_k$.
Choose a step $\tilde{s}_k$ that (sufficiently) reduces the model $m_k(x_{k+1}, y_k)$ such that $(x_{k+1}, y_k) + \tilde{s}_k \in B_k(x_k; y_k; \Delta_k)$. Note y **is not fixed**
Set $(x_{k+1}, y_{k+1}) = (x_{k+1}, y_k) + \tilde{s}_k$

Step 6: Trust-region radius update. Set

$$\Delta_{k+1}^{icb} \in \begin{cases} [\Delta_k, \min\{\gamma_{inc}\Delta_k, \Delta_{max}\}] & \text{if } \rho_k \geq \eta_1, \\ \{\gamma\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \text{ } \textit{is} \text{ FL}, \\ \{\Delta_k\} & \text{if } \rho_k < \eta_1 \text{ and } m_k \\ & \text{is not certifiably fully linear.} \end{cases}$$

Increment k by one and go to Step 1.

Note: the model-improvement algorithm to improve the model until it is fully linear on the required trust region can be done in a finite, uniformly bounded number of steps .

Salient points

In general terms the basis for convergence is:

1. Do at least as well as (a fixed fraction of) the Cauchy point. (the minimum of the model in the "steepest descent" direction within the trust region)
2. Manage the size of the trust region (delta) appropriately
3. Have consistency between F and m
4. Because of **3** we want to define the Cauchy point, guarantee the convergence, and do the trust region management, etc for **fixed** y
5. Solve the subproblem (now the y is **not fixed**) using, for example Bonmin. Its solution should eventually be better than the value for the Cauchy point for the fixed y.

Iterating, eventually will have the right y and converge to the solution!

Salient points

In general terms the basis for convergence is:

1. Do at least as well as (a fixed fraction of) the Cauchy point. (the minimum of the model in the "steepest descent" direction within the trust region)
2. Manage the size of the trust region (delta) appropriately
3. Have consistency between F and m
4. Because of **3** we want to define the Cauchy point, guarantee the convergence, and do the trust region management, etc for **fixed** y
5. Solve the subproblem (now the y is **not fixed**) using, for example Bonmin. Its solution should eventually be better than the value for the Cauchy point for the fixed y.

Iterating, eventually will have the right y and converge to the solution!

Salient points

In general terms the basis for convergence is:

1. Do at least as well as (a fixed fraction of) the Cauchy point. (the minimum of the model in the "steepest descent" direction within the trust region)
2. Manage the size of the trust region (delta) appropriately
3. Have consistency between F and m
4. Because of 3 we want to define the Cauchy point, guarantee the convergence, and do the trust region management, etc for **fixed** y
5. Solve the subproblem (now the y is **not fixed**) using, for example Bonmin. Its solution should eventually be better than the value for the Cauchy point for the fixed y.

Iterating, eventually will have the right y and converge to the solution!

Salient points

In general terms the basis for convergence is:

1. Do at least as well as (a fixed fraction of) the Cauchy point. (the minimum of the model in the "steepest descent" direction within the trust region)
2. Manage the size of the trust region (delta) appropriately
3. Have consistency between F and m
4. Because of **3** we want to define the Cauchy point, guarantee the convergence, and do the trust region management, etc for **fixed** y
5. Solve the subproblem (now the y is **not fixed**) using, for example Bonmin. Its solution should eventually be better than the value for the Cauchy point for the fixed y.

Iterating, eventually will have the right y and converge to the solution!

Salient points

In general terms the basis for convergence is:

1. Do at least as well as (a fixed fraction of) the Cauchy point. (the minimum of the model in the "steepest descent" direction within the trust region)
2. Manage the size of the trust region (delta) appropriately
3. Have consistency between F and m
4. Because of **3** we want to define the Cauchy point, guarantee the convergence, and do the trust region management, etc for **fixed** y
5. Solve the subproblem (now the y is **not fixed**) using, for example Bonmin. Its solution should eventually be better than the value for the Cauchy point for the fixed y.

Iterating, eventually will have the right y and converge to the solution!

Numerical Results

Solved about 60 test problems for which optima could be directly verified (<= 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results

Solved about 60 test problems for which optima could be directly verified (<= 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results

Solved about 60 test problems for which optima could be directly verified ($<=$ 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results

Solved about 60 test problems for which optima could be directly verified ($<=$ 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results

Solved about 60 test problems for which optima could be directly verified ($<=$ 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results

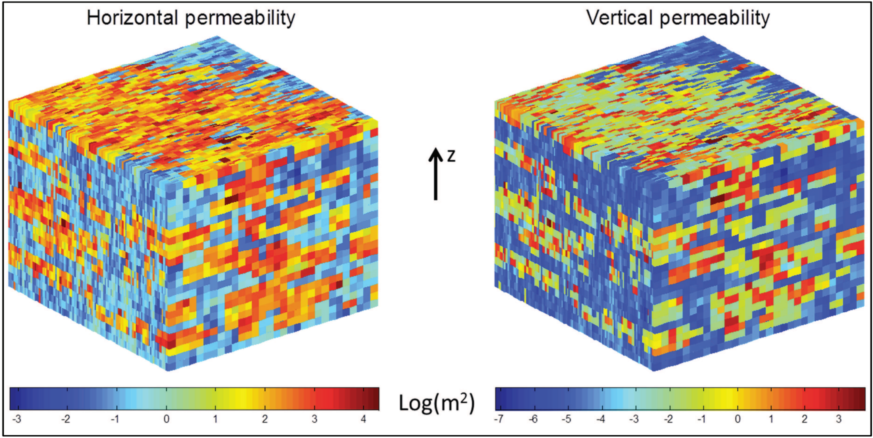
Solved about 60 test problems for which optima could be directly verified ($<=$ 10 variables of each type).

- Typically did as well as Bonmin or better
- Always reaches a local solution and usually very fast
- Can do worse than SCIP **when scip converges**
- Consistently much better than NOMAD as one would expect
- But remember that there are discrete issues.
- And we still have plenty of work to do!.

Numerical Results: History Matching

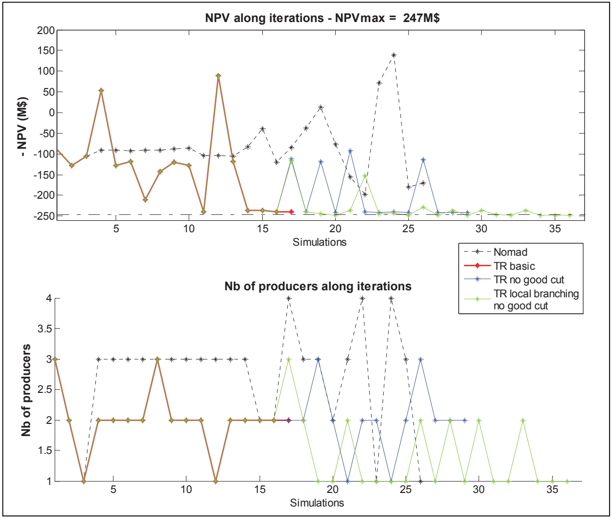
50 layers of $2'$ with 60×220 cells $20' \times 10'$
Up-scaled to $30 \times 110 \times 25$ cells of $80' \times 40' \times 4'$
10 yrs production: 1 injector well, 1 — 4 producers.

Optimize the number of wells and their locations to maximize the NPV of the field.



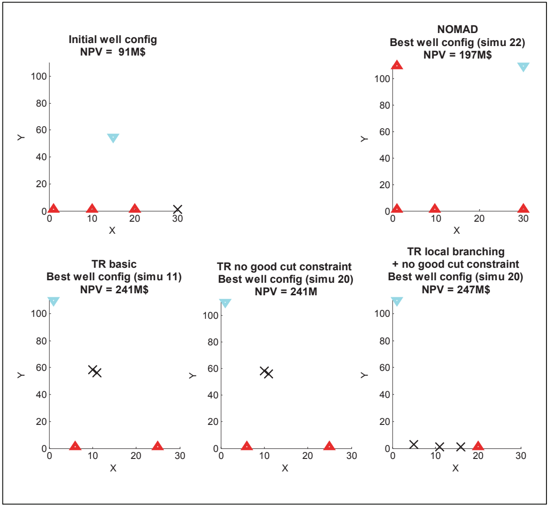
Numerical Results (continued)

Number of variables being set is 14 continuous and 4 binary variables



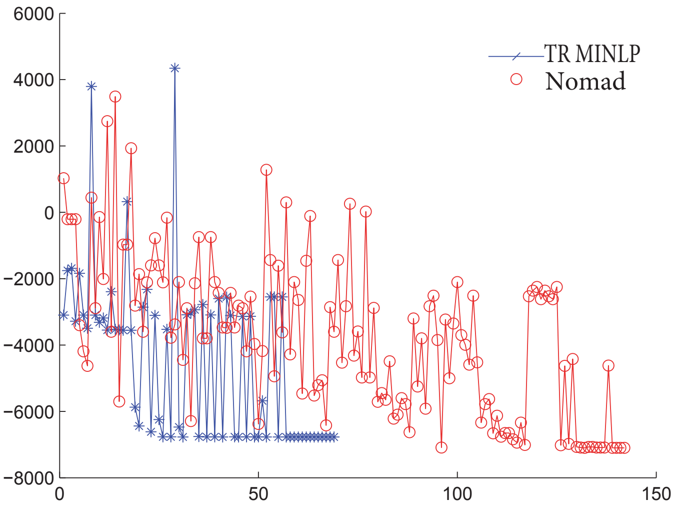
Numerical Results Compare NOMAD solutions & ours

Run with 3 different tunings. The initial configuration is displayed at top left.



Numerical Results (continued)

Number of variables being set is 4 continuous and 8 binary variables



Numerical Results (continued)

Number of variables being set is 4 continuous and 8 binary variables