Methods for Local and Global Optimization

Constantinos Voglis
vogliscs.uoi.gr

Computer Science Department, University of Ioannina

June 7, 2010

Computer Science Department University of Ioannina

1. Presentation Outline
2. Introductory material
3. Optimality Conditions

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - **Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.**
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - **Tremendous growth in computing power that we have witnessed in our times.**
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - **finding molecular conformation;**
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - **finding the optimal trajectory for an aircraft or a robot arm;**
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - **identifying the seismic properties;**
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - **designing a portfolio to maximize expected return;**
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - **controlling a chemical process or a mechanical device to optimize performance;**
  - computing the optimal shape of an automobile or aircraft component;
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - **computing the optimal shape of an automobile or aircraft component;**
  - identifying parameters in machine learning problems

Introduction To Optimization

- In recent years, the field of Optimization, has undergone a rapid development:
  - Optimization applications in areas science and technology, including molecular biology, imaging, digital signal processing, portfolio management, networks and more.
  - Tremendous growth in computing power that we have witnessed in our times.
- Applications:
  - finding molecular conformation;
  - finding the optimal trajectory for an aircraft or a robot arm;
  - identifying the seismic properties;
  - designing a portfolio to maximize expected return;
  - controlling a chemical process or a mechanical device to optimize performance;
  - computing the optimal shape of an automobile or aircraft component;
  - **identifying parameters in machine learning problems**

Mathematical Formulation

Optimization is the minimization or maximization of a function subject to constraints on its variables.

- **x** is the vector of **n** variables, also called unknowns or parameters;
- **f** is the objective function, a function of x that we want to maximize or minimize;
- **S** a compact subset of R^n.

The optimization problem may be stated as:

Optimization Problem

$$\min_{x \in \mathcal{R}^n} f(x) \quad \text{subject to } x \in S \subset R^n$$

Optimization Taxonomy

- Optimization parameters (variables)
  - **Continuous Optimization**
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - **Discrete Optimization**
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - **Unconstrained Optimization ($S = R^n$)**
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - **Constrained Optimization**
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - **Local Optimization**
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - **Global Optimization**
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - **Stochastic**
  - Deterministic
    - Continuous
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - **Deterministic**
    - **Continuous**
    - **Differentiable**
    - **Non-Smooth**

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - **Continuous**
    - Differentiable
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - **Differentiable**
    - Non-Smooth

Optimization Taxonomy

- Optimization parameters (variables)
  - Continuous Optimization
  - Discrete Optimization
- Search space S
  - Unconstrained Optimization ($S = R^n$)
  - Constrained Optimization
- Quality of solution
  - Local Optimization
  - Global Optimization
- Objective function
  - Stochastic
  - Deterministic
    - Continuous
    - Differentiable
    - **Non-Smooth**

Commonly used Notations

Objective Function Related

$f(x)$	:	objective function
$g(x), \nabla f(x)$	:	objective function's gradient (first order derivatives)
$H(x), \nabla^2 f(x)$	:	objective function's Hessian matrix (second order derivatives)
$J(x), \dfrac{\nabla f_i(x)}{\nabla x_j}$	:	objective function's Jacobian matrix (sum of squares)

Optimization related

x^*	:	a minimum (local or global)
λ, μ	:	Lagrange multipliers
$L(x; \lambda)$	:	Lagrangian function

Definitions

Part I

Global Optimization

Presentation Outline

4. Topographically adapted stochastic search
5. Sampling from sum of normals distribution
6. Spectral information based Clustering
7. A new strictly descent local search
8. A new stopping rule

Introduction - Notation

Global optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to :} \quad & x \in S \subset R^n \end{aligned}$$

Generalized global optimization problem

$$\begin{aligned} \text{Find all minima} \quad & f(x) \\ \text{subject to :} \quad & x \in S \subset R^n \end{aligned}$$

- Computational Physics (few-body systems, optical systems)
- Computational Chemistry (drug design)
- Radiation therapy
- Model fitting (neural network training)
- Molecular conformation

Global optimization taxonomy

Global optimization methods are divided:

- Stochastic vs. Deterministic
- Continuous vs. Discrete
- Single global vs. Multiple / all global
- Heuristic vs. Meta-heuristic
- With local optimization (two-phase) vs. Only global phase

In this thesis we are concerned:

Continuous: Function (search space) and margins
Stochastic: Random sampling
Many minima: We seek all minima in a specified domain
Two-phase: Local optimization application .

Stochastic, two-phase, clustering

Algorithmic framework 1

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. Cluster analysis: Group sampled points and assign them to minima.

S3. Local search: Apply a local search from a representative point of each cluster.

S4. Stopping rule: Decide whether to stop or continue.

Contribution

Stochastic, two-phase, clustering

Algorithmic framework 1

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. **Cluster analysis: Group sampled points and assign them to minima.**

S3. Local search: Apply a local search from a representative point of each cluster.

S4. Stopping rule: Decide whether to stop or continue.

Contribution

Application of spectral clustering+global k-means

Stochastic, two-phase, clustering

Algorithmic framework 1

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. **Cluster analysis: Group sampled points and assign them to minima.**

S3. **Local search: Apply a local search from a representative point of each cluster.**

S4. Stopping rule: Decide whether to stop or continue.

Contribution

Application of spectral clustering+global k-means

New strictly descent local search

Stochastic, two-phase, clustering

Algorithmic framework 1

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. **Cluster analysis: Group sampled points and assign them to minima.**

S3. **Local search: Apply a local search from a representative point of each cluster.**

S4. **Stopping rule: Decide whether to stop or continue.**

Contribution

Application of spectral clustering+global k-means

New strictly descent local search

New stopping criterion based on uniform minima distribution

Stochastic, two-phase, adaptive distribution

Contribution

General Algorithmic Framework 2

S1. Sample from adaptive distribution: Of Implicit or explicit form.

S2. Apply local search: Same as previous framework

S3. Update distribution parameters: From the minima retrieved so far.

S4. Stopping rule: Same as previous framework

Stochastic, two-phase, adaptive distribution

General Algorithmic Framework 2

S1. **Sample from adaptive distribution: Of Implicit or explicit form.**

S2. Apply local search: Same as previous framework

S3. **Update distribution parameters: From the minima retrieved so far.**

S4. Stopping rule: Same as previous framework

Contribution

1. Implicit sampling: Topologically adaptive method
2. Explicit sampling: Sum of normal distributions

Stochastic, two-phase, adaptive distribution

General Algorithmic Framework 2

S1. **Sample from adaptive distribution: Of Implicit or explicit form.**

S2. **Apply local search: Same as previous framework**

S3. **Update distribution parameters: From the minima retrieved so far.**

S4. Stopping rule: Same as previous framework

Contribution

1. Implicit sampling: Topologically adaptive method
2. Explicit sampling: Sum of normal distributions

New strictly descent local search

Stochastic, two-phase, adaptive distribution

General Algorithmic Framework 2

S1. **Sample from adaptive distribution: Of Implicit or explicit form.**

S2. **Apply local search: Same as previous framework**

S3. **Update distribution parameters: From the minima retrieved so far.**

S4. **Stopping rule: Same as previous framework**

Contribution

1. Implicit sampling: Topologically adaptive method
2. Explicit sampling: Sum of normal distributions

New strictly descent local search

New stopping criterion based on uniform minima distribution

Topographically adapted stochastic search

Method properties:

- Decides whether to start a local search or not.
- Aims one local search per minimum.
- Stores local minima and information about them.
- Defines a spherical model around every minimum.
- Asymptotic guarantee
- Can be considered as implicit sampling distribution!

Region of attraction

The **region of attraction** of a local minimum associated with a local search procedure $\mathcal{L}$ is defined as:

$$A_i \equiv \{x \in S, \mathcal{L}(x) = x_i^*\}$$

If S contains a total of w local minima, from the definition above follows:

$$\cup_{i=1}^{w} A_i = S$$

$$m(S) = \sum_{i=1}^{w} m(A_i) \quad \text{for deterministic local search}$$

If K points are sampled from S, the apriori probability that at least one point is contained in A_i is given by:

$$1 - \left(1 - \frac{m(A_i)}{m(S)}\right)^K = 1 - (1 - p_i)^K$$

Ideal algorithm

Imagine an ‘Ideal two-phase algorithm-like Algorithm:

S1 Sample: Sample $x \in S$

S2 Main step: If $(x \notin \cup_{i=1}^{k} A_i)$ Then

$$y = \mathcal{L}(x)$$

$$k = k + 1$$

$$y_k = y$$

Endif

S3 Termination Control: If a stopping rule applies, STOP.

Ideal algorithm

Imagine an ‘Ideal two-phase algorithm-like Algorithm:

S1 Sample: Sample $x \in S$

S2 Main step: If $(x \notin \cup_{i=1}^{k} A_i)$ Then

$$y = \mathcal{L}(x)$$
$$k = k + 1$$
$$y_k = y$$

Endif

S3 Termination Control: If a stopping rule applies, STOP.

a) Every minimum is located exactly once.
b) We assume that the region of attraction may be directly determined.

Practical implementation

- Since the regions of attraction of the minima discovered so far, are not known, it is not possible to determine if a point belongs or not to their union.
- However, a probability may be estimated, based on several assumptions.
- Hence, a stochastic modification may render IMS useful.

Stochastic modification of the main step:

S2 Main step:

Calculate the probability p_{local}, that $x \notin \cup_{i=1}^{k} A_i$

Draw a random number $\xi \in (0, 1)$ from a uniform distribution

If ($\xi < p_{local}$) Then

$y = \mathcal{L}(x)$

If ($y \notin \{y_i, i = 1, 2, \ldots, k\}$) Then

$k = k + 1$

$y_k = y$

Else

Update information (R_i, l_i)

Endif

Endif

Practical implementation

- **Since the regions of attraction of the minima discovered so far, are not known, it is not possible to determine if a point belongs or not to their union.**
- However, a probability may be estimated, based on several assumptions.
- Hence, a stochastic modification may render IMS useful.

Stochastic modification of the main step:

S2 Main step:

Calculate the probability p_{local}, that $x \notin \cup_{i=1}^{k} A_i$

Draw a random number $\xi \in (0, 1)$ from a uniform distribution

If ($\xi < p_{local}$) Then

  $y = \mathcal{L}(x)$

  If ($y \notin \{y_i, i = 1, 2, \ldots, k\}$) Then

    $k = k + 1$

    $y_k = y$

  Else

    Update information (R_i, l_i)

  Endif

Endif

Practical implementation

- Since the regions of attraction of the minima discovered so far, are not known, it is not possible to determine if a point belongs or not to their union.
- **However, a probability may be estimated, based on several assumptions.**
- Hence, a stochastic modification may render IMS useful.

Stochastic modification of the main step:

S2 Main step:

Calculate the probability p_{local}, that $x \notin \cup_{i=1}^{k} A_i$

Draw a random number $\xi \in (0, 1)$ from a uniform distribution

If ($\xi < p_{local}$) Then

    $y = \mathcal{L}(x)$

    If ($y \notin \{y_i, i = 1, 2, \ldots, k\}$) Then

        $k = k + 1$

        $y_k = y$

    Else

        Update information (R_i, l_i)

    Endif

Endif

Practical implementation

- Since the regions of attraction of the minima discovered so far, are not known, it is not possible to determine if a point belongs or not to their union.
- However, a probability may be estimated, based on several assumptions.
- **Hence, a stochastic modification may render IMS useful.**

Stochastic modification of the main step:

S2 Main step:

Calculate the probability p_{local}, that $x \notin \cup_{i=1}^{k} A_i$

Draw a random number $\xi \in (0, 1)$ from a uniform distribution

If ($\xi < p_{local}$) Then

$y = \mathcal{L}(x)$

If ($y \notin \{y_i, i = 1, 2, \ldots, k\}$) Then

$k = k + 1$

$y_k = y$

Else

Update information (R_i, l_i)

Endif

Endif

Practical implementation

- Since the regions of attraction of the minima discovered so far, are not known, it is not possible to determine if a point belongs or not to their union.
- However, a probability may be estimated, based on several assumptions.
- Hence, a stochastic modification may render IMS useful.

Stochastic modification of the main step:

S2 Main step:

Calculate the probability p_{local}, that $x \notin \cup_{i=1}^{k} A_i$

Draw a random number $\xi \in (0, 1)$ from a uniform distribution

If ($\xi < p_{local}$) Then

$y = \mathcal{L}(x)$

If ($y \notin \{y_i, i = 1, 2, \ldots, k\}$) Then

$k = k + 1$

$y_k = y$

Else

Update information (R_i, l_i)

Endif

Endif

Probability estimation

- Estimate p, that $x \notin \cup_{i=1}^{k} A_i$.
- Overestimated probability ($p > 1$), increases the computational cost, and transforms the algorithm towards the standard MultiStart.
- Underestimated probability will cause an iteration delay without significant computational cost. (Only sampling, no local search).

Probability model

If a sample point x is close to an already known minimizer y_i, the probability that it does not belong to its region of attraction is small and zero at the limit of complete coincidence.

$$Pr(x \notin A_i) \xrightarrow[|x-y_i| \longrightarrow 0]{} 0$$

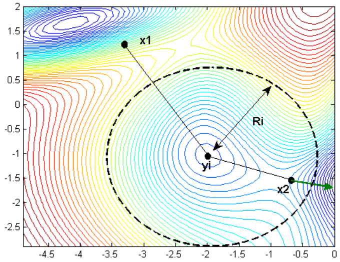
Probability model

Let us define the *maximum attractive radius (MAR)* as:

$$R_i = \max_j \{||x_j^{(i)} - y_i||\}$$

where $x_j^{(i)}$ are the sampled points which led the subsequent local search to the i^{th} minimizer y_i.

- If $||y_i - x|| < R_i$, then x is likely to be inside the region of attraction of y.
  - If however $\nabla f(x)^T(y_i - x) \geq 0$, i.e. the direction from x to y_i is ascent.
- If $||y_i - x|| > R_i$, then $Pr(x \notin A_i) = 1$



Probability model

Probability model

$$Pr(x \notin A(y)) = \begin{cases} 1, \text{ if } z > 1 \text{ or } \nabla f(x)^T (y-x) \geq 0 \\ \phi(z,l) \times \left[1 + \frac{(y-x)^T \nabla f(x)}{z||\nabla f(x)||}\right], \text{otherwise} \end{cases}$$

$z = \frac{||y_i - x||}{r_i}$, l is the number of times y has been recovered so far

$\phi(z,l)$ has

$$\lim_{z \to 0} \phi(z,l) \to 0$$

$$\lim_{z \to 1} \phi(z,l) \to 1$$

$$\lim_{l \to \infty} \phi(z,l) \to 0$$

$$0 < \phi(z,l) < 1$$

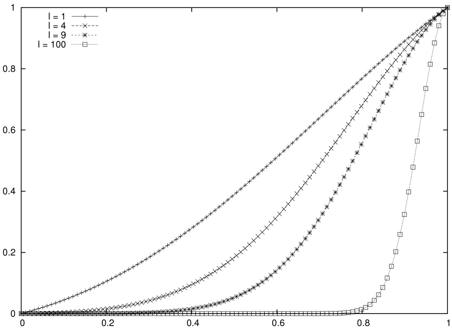
$p_g = \left[1 + \frac{(y-x)^T \nabla f(x)}{z||\nabla f(x)||}\right]$ is a reducing factor s.t.

$$p_g \to 0 \text{ as } \frac{(y-x)^T \nabla f(x)}{z||\nabla f(x)||} \to -1$$

(perpendicular)

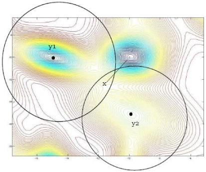
A model for $\phi(z, l)$

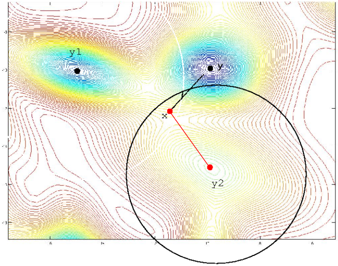
$$\phi(z,l) = z e^{-l^2(z-1)^2},\ \forall z \in (0,1)$$



To start or not to start...

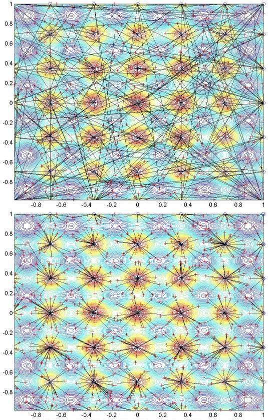
- Goal: Calculate $p_{local} = Pr(x \notin \cup_{i=1}^{k} A_i)$
- or $p_{local} = \prod_{i=1}^{k} Pr(x \notin A_i)$
- Approximate: $\tilde{p}_{local} = Pr(x \notin \cup_{i=1}^{k} A_k)$, A_n being the region of attraction of the nearest to x discovered minimizer y_k.





Local search properties

- The probability model is based on distances from the discovered minima.
  - One model per minimum!
- It is implicitly assumed that the closer to a minimum a point is, the greater the probability that falls inside its Region of Attraction.
- This is not true for all local search procedures L.
- Regions of attraction should contain the minimum and be contiguous.
- Ideally the regions of attraction should resemble the ones produced by a descent method with infinitesimal step.



Asymptotic guarantee

Adapt Algorithm

Sample: Sample $x \in S$

Main step: i = $\underset{j=1,\ldots,k}{argmin} ||x - y_j||$

$d = ||x - y_i||$

If $(d < r_i)$ Then

If $(\nabla f(x)^T (y_i - x) < 0)$ Then

$$z = \frac{||y_i - x||}{r_i}$$

$$p = \phi(z, n_i)\left[1 + \frac{(y_i - x)^T \nabla f(x)}{||(y_i - x)^T \nabla f(x)||}\right]$$

Else

$p = 1.0$

Endif

Else

$p = 1.0$

Endif

Let ξ be a uniform random in $[0, 1]$

If $(\xi < p)$ Then

Local Search: $y = \mathcal{L}(x)$

If (y is new minimum) Then

$k = k + 1,\ r_k = ||x - y_k||,\ n_k = 1$

Else { *We discovered the l-th local minimum* }

$r_l = \max(r_l, ||x - y_l||),\ n_l = n_l + 1$

Endif

Else { *Assuming that x belongs in the region of attraction of the i-th minimum* }

$r_i = \max(r_i, ||x - y_i||),\ n_i = n_i + 1$

Endif

Termination Control: If a stopping rule applies, STOP.

Sampling from sum of normals distribution

Recall the **General Algorithmic Framework 2**

S1. Sample from adaptive distribution: Of Implicit or explicit form.

S2. Apply local search: Same as previous framework

S3. Update distribution parameters: From the minima retrieved so far.

S4. Stopping rule: Same as previous framework

Proposed method's properties:

- Explicit definition of sampling distribution.
- One model (normal distribution) per minimum.
- Rejection sampling scheme
- Computationally intensive.

Sampling from sum of normals distribution

Recall the **General Algorithmic Framework 2**

S1. **Sample from adaptive distribution: Of Implicit or explicit form.**

S2. Apply local search: Same as previous framework

S3. **Update distribution parameters: From the minima retrieved so far.**

S4. Stopping rule: Same as previous framework

Proposed method's properties:

- Explicit definition of sampling distribution.
- One model (normal distribution) per minimum.
- Rejection sampling scheme
- Computationally intensive.

Sum of normals distribution model (I)

Let y_i be a local minimum:

$$N(x;\mu_{y_i},\Sigma_{y_i}) = \frac{1}{\sqrt{2\pi}}\frac{1}{|\Sigma_{y_i}|}e^{-\frac{1}{2}(x-\mu_{y_i})^T\Sigma_{y_i}^{-1}(x-\mu_{y_i})}$$

where μ_{y_i} = mean value and Σ_{y_i} = covariance matrix of the distribution. Given N points $x_1, x_2, \ldots x_N$ that lead to local minimum y_i we can calculate the mean value μ and the covariance matrix Σ using **maximum likelihood estimates**:

$$\begin{aligned}
\mu_{y_i} &= E(X) = \frac{1}{N}\sum_{i=1}^{N} x_i \\
\Sigma_{y_i} &= E\left((X-\mu_{y_i})(X-\mu_{y_i})^T\right) = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu_{y_i})(x_i-\mu_{y_i})^T
\end{aligned}$$

Sum of normals distribution model(II)

Consider w minima:

$$\mathcal{N}(x) = \sum_{i=1}^{w} \eta_i N(x; \mu_{y_i}, \Sigma_{y_i})$$

where $\eta_i = \frac{\rho_i}{\sum \rho_i}$, ρ_i the number of local searches reached y_i. Proposed: Sample using $\mathcal{N}(x)$.

$$\mathcal{N}(x) \geq 0, \quad \forall x$$
$$\int \mathcal{N}(x)dx = 1$$

Sampling

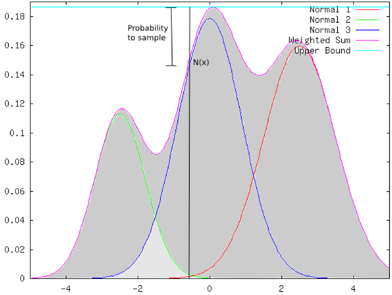
Rejection sampling from function $f(x)$

Consider an instrumental distribution function $g(x)$

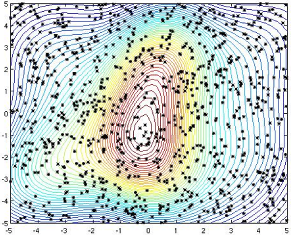
* Sample x from $g(x)$ and u from $U(0,1)$
* Check whether or not $u < \frac{f(x)}{Mg(x)}$.
    - If this holds, accept x as a realization of $f(x)$;
    - if not, reject the value of x and repeat the sampling step.

Sampling Algorithm

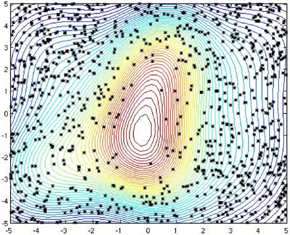
- repeat the following until a point is accepted
  - Get x from $U([a,b]^n)$ where n is the problem's dimension.
  - Sample ξ from $U(0,1)$
  - $\tilde{F} \leftarrow 0$, $maxF \leftarrow 0$
  - for every local minimum retrieved
    - $\tilde{F} \leftarrow \tilde{F} + \frac{\rho_i}{\sum_j \rho_j} N(x, \mu_{y_i}, \Sigma_{y_i})$
    - $maxF \leftarrow maxF + \frac{rho_i}{\sum_j \rho_j} N(\mu_{y_i}, \mu_{y_i}, \Sigma_{y_i})$
  - If $\xi \times maxF > \tilde{F}$ then accept x as starting point



Illustrative Example (Single minimum)

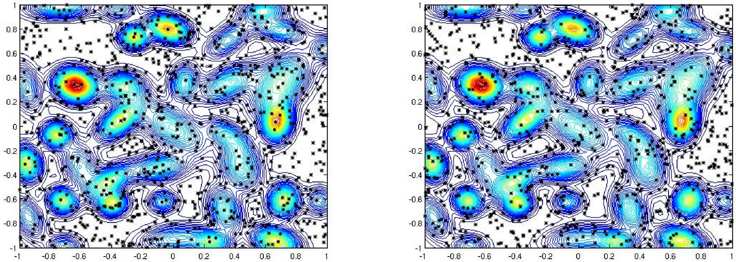


(a) Uniform distribution



(b) Proposed distribution

Illustrative Example (Multiple minima)



(c) Uniform distribution

(d) Proposed distribution

Sequential parameter update

Let starting point x s.t. $\mathcal{L}(x) = y_i$

- $\tilde{\mu}_{y_i} \leftarrow \mu_{y_i} + \alpha_i(x - \mu_{y_i})$
- $\tilde{\Sigma}_{y_i} \leftarrow \Sigma_{y_i} + \alpha_i(x - \mu_{y_i})(x - \mu_{y_i})^T$,

where $\alpha_i \in (0, 1)$.

Quantity α_i is called learning factor:

- $\alpha_i = \frac{\rho_i}{\sum \rho_i}$, ρ_i how many times y_i is found
- α_i predefined constant

Initialization:

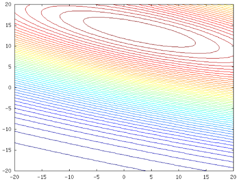
- Initialize μ_{y_i}
  1. $\mu_{y_i} = (x_{first} - y_i)/2$
  2. $\mu_{y_i} = y_i$
- Initialize Σ_{y_i}
  1. $\Sigma_{y_i} = \sigma I_n$
  2. Initialize using hessian matrix at the minimum

Cholesky factorization

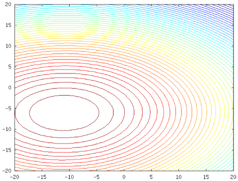
For numerical stability and computational efficiency we use Cholesky factor $\Sigma = LL^T$:

- Cheaper update $O(n^2)$: $\tilde{\Sigma} = \Sigma + uu^T = LL^T + uu^T = L(I + pp^T)L^T$, where $Lp = u$.
- Efficient determinant calculation: $det(\Sigma) = \prod_{i=1}^{N} L_{ii}^2$
- Efficient exponential term calculation: $(x-\mu)^T\Sigma^{-1}(x-\mu) = \left(L^{-1}(x-\mu)\right)^T \left(L^{-1}(x-\mu)\right)$, where $L^{-1}(x-\mu) = y \Rightarrow Ly = x - \mu$

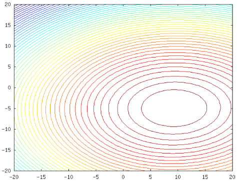
Update example (Single minimum)



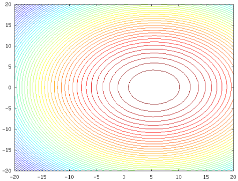
(a)



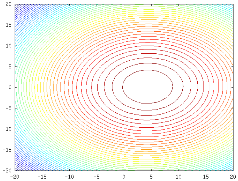
(b)



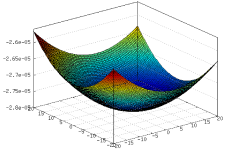
(c)



(d)



(e)



(f)

Computational resuls

Spectral information based Clustering

From **Algorithmic framework 1**

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. Cluster analysis: Group sampled points and assign them to minima.

S3. Local search: Apply a local search from a representative point of each cluster.

S4. Stopping rule: Decide whether to stop or continue.

Method's properties:

- Novel clustering approach:
  - spectral clustering
  - global k-means
- Each cluster represents a minimum
- Apply local search from cluster center
- Basic computational cost, eigenvalue decomposition

Spectral information based Clustering

From **Algorithmic framework 1**

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. **Cluster analysis: Group sampled points and assign them to minima.**

S3. Local search: Apply a local search from a representative point of each cluster.

S4. Stopping rule: Decide whether to stop or continue.

Method's properties:

- Novel clustering approach:
  - spectral clustering
  - global k-means
- Each cluster represents a minimum
- Apply local search from cluster center
- Basic computational cost, eigenvalue decomposition

Clustering in global optimization

- Hierarchical clustering
  - Density clustering
  - Single linkage clustering
  - Multilevel single linkage clustering)
- Partitional clustering
  - Mode seeking algorithm
  - Multilevel mode seeking
  - Vector quantization

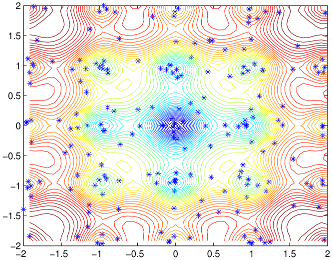
Proposed clustering overview

Let random sampled points in search space

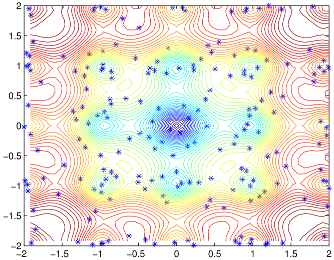
- Step 1(Concentrate points): Move sampled points toward the closest minimum
- Step 2(Estimate number of clusters k): Estimate k using spectral information from an affinity matrix and including gradient information
- Step 3(Apply clustering): Apply global k-means either on the problem's space or on spectral space
- Step 4(Retrieve minima): Start appropriate local optimization

Form clusters using:

- Small steps in negative gradient direction (parameter depended)
- Few iterations of a local search



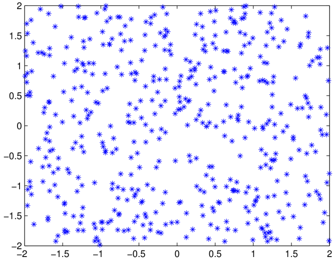
(a) Small steps in negative gradient direction



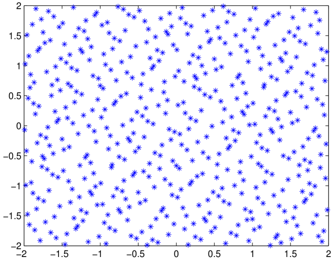
(b) Few iteration of a local search

Uniform vs. pseudo-uniform

A clear choice instead of uniform could be a pseudo-uniform (eg. Halton sequence)



(a) Uniform



(b) Pseudo-uniform

Estimate k: Spectral analysis

Spectral analysis algorithm

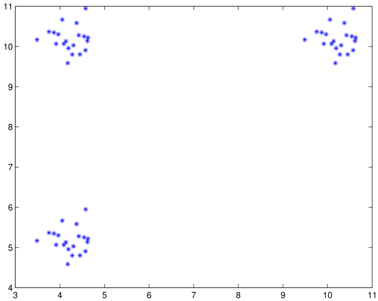
Let M concentrated random sampled points.

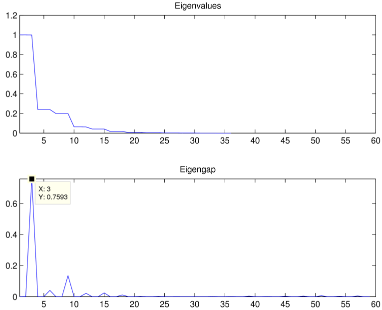
1. Construct affinity matrix $A \in R^{N \times N}$ where $A_{ij} = exp(-||x_i - x_j||^2/2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$
2. Define diagonal matrix D where element (i, i) is the sum of the i-th row of A
3. Construct $L = D^{-1/2}AD^{-1/2}$
4. Calculate and sort the eigenvalues of L. Let $e_1, e_2, \ldots, e_N$ be the sorted eigenvalues
5. Calculate differences $\delta_i = e_{i+1} - e_i, i = 1, \ldots, N-1$.
6. Find the maximum difference (Eigengap): $k = argmax_{k=1,\ldots,N-1}\delta_i$

Maximum difference (Eigengap) of the eigenvalues of L

- Origins in graph partitioning)
- Number of connected components $\equiv$ number of eigenvalues close to 1
- Alternative criterion 1: Define a threshold (say < 0.7)
- Alternative criterion 2: Locate the first eigenvalue < 1

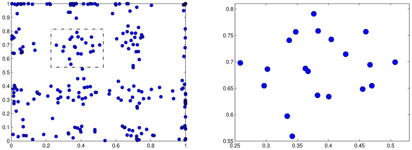
Example toy problem





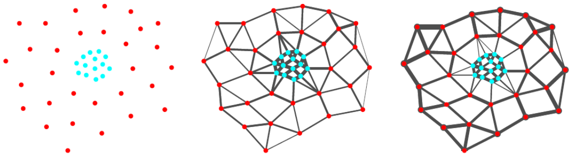
σ parameter calculation

Parameter σ is used in $A_{ij} = exp(-||x_i - x_j||^2/2\sigma^2)$.



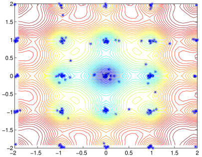
σ calculation

σ is defined from the mean distance from a point to its l_{nei} closest neighbors (local scaling).

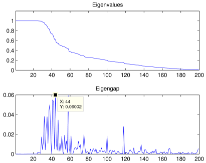


Parameter σ calculation: l_{nei}

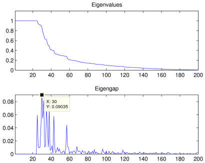
The number of neighbors l_{nei} plays important role in for the calculation of σ.



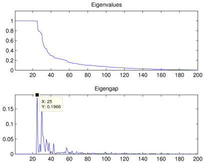
(a) Well concentrated samples



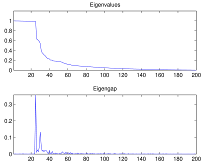
(b) $l_{nei} = 2$



(c) $l_{nei} = 3$



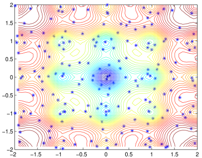
(d) $l_{nei} = 4$



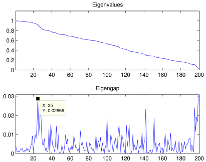
(e) $l_{nei} = 5$

Parameter σ calculation: l_{nei}

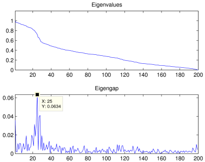
The number of neighbors l_{nei} plays important role in for the calculation of σ.



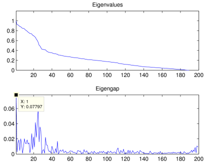
(a) Slightly transformed samples



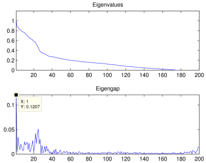
(b) $l_{nei} = 2$



(c) $l_{nei} = 3$



(d) $l_{nei} = 4$



(e) $l_{nei} = 5$

Gradient information

- Matrix A stores the affinity between two points x_i and x_j
- The affinity is based on Euclidean distance $exp(-||x_i - x_j||^2/2\sigma^2)$

We propose an additional information in matrix A that includes gradient information: *Two points are correlated if following their negative gradients results in smaller distance*

Gradient check x_i and x_j

Let $dx = x_i - x_j$ and $dg = \nabla f(x_i) - \nabla f(x_j)$ then if $\omega = \dfrac{dx^T dg}{||dx||||dg||} < 0$ the points are correlated.

Calculate A

$$A_{i,j} = \begin{cases} exp(-||x_i - x_j||^2/2\sigma^2 & \text{if } \omega < 0 \\ 0 & \text{otherwise} \end{cases}$$

Gradient information: Illustration 1

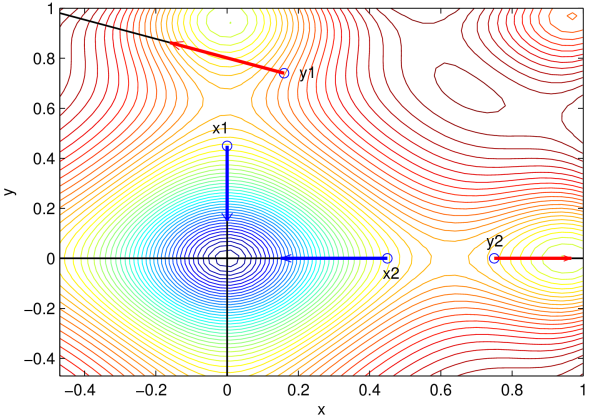
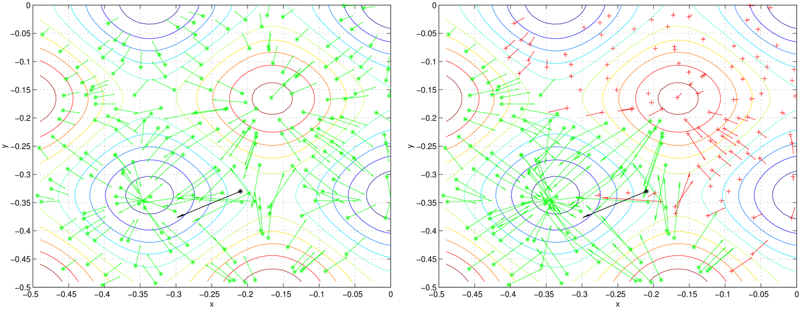


Figure: Example of association and disassociation using the gradient

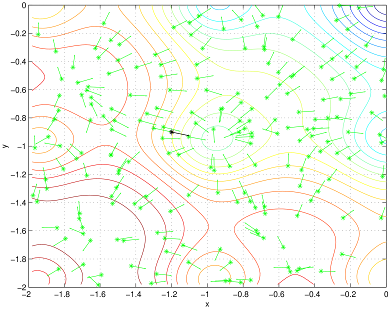
Gradient information: Illustration 2



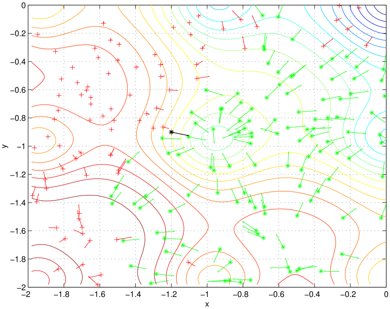
(a) Plot pairwise affinities: Without gradient

(b) Plot pairwise affinities: With gradient

Gradient information: Illustration 2

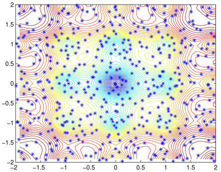


(a) Plot pairwise affinities: Without gradient

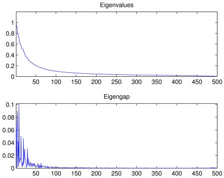


(b) Plot pairwise affinities: With gradient

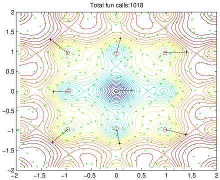
Gradient information: Impact on clustering



(a) Sampled points

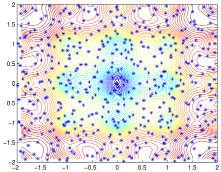


(b) Without derivative - eigenvalues

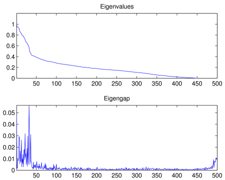


(c) Without derivative - clusters

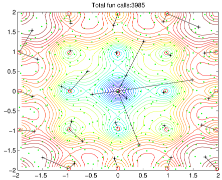
Gradient information: Impact on clustering



(a) Sampled points



(b) With derivative - eigenvalues



(c) With derivative - clusters

Global k-means

The global k-means algorithm:

- Needs the number of clusters k
- Partitions the dataset incrementally to $j = 1, 2, \dots, k$ clusters
- Uses the information of j-th step (j clusters) to construct $j + 1$-th step($j + 1$ clusters)
- In the j-th step performs M times the simple k-means algorithm
- Independent of initialization

Extension

Global k-means can operate on affinity matrix by defining *medoids* instead of of means.

Global k-means animation

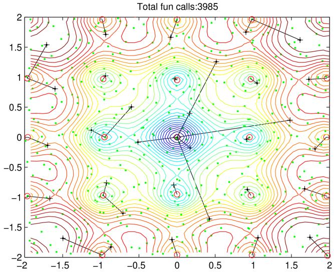
Global k-means / Global k-means on affinity matrix

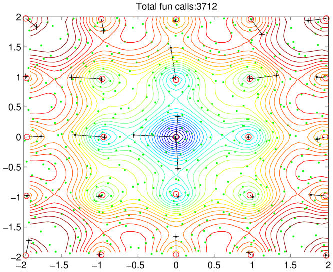
Global k-means:

- **Operates using distances**
- No use of gradient information
- Cluster centers are new points

Global k-means on affinity matrix:

- **Operates on affinity matrix**
- Uses gradient information stored in affinity matrix
- Cluster centers are existing points





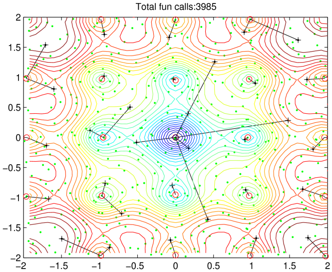
Global k-means / Global k-means on affinity matrix

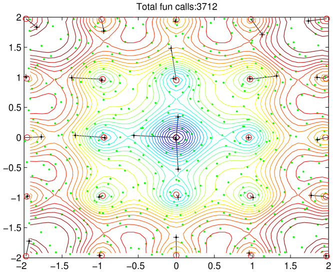
Global k-means:

- Operates using distances
- **No use of gradient information**
- Cluster centers are new points

Global k-means on affinity matrix:

- Operates on affinity matrix
- **Uses gradient information stored in affinity matrix**
- Cluster centers are existing points





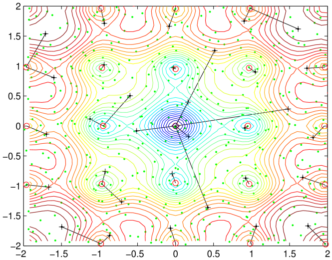
Global k-means / Global k-means on affinity matrix

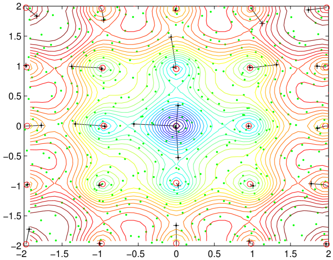
Global k-means:

- Operates using distances
- No use of gradient information
- **Cluster centers are new points**

Global k-means on affinity matrix:

- Operates on affinity matrix
- Uses gradient information stored in affinity matrix
- **Cluster centers are existing points**



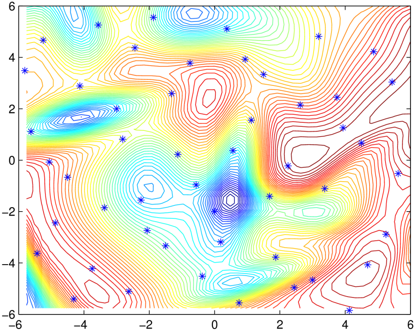


Final algorithm

Input: **f**: Minimizing function, **xl, xu**: Problem's bounds, **N**: Sample size, **Isampl:** Defines sample strategy, **Ired**: Switches between the concentrating method, **Iuseg:** Use gradient information, **Iclust**: Switches between global kmeans/kmedoids, **Inei**: Number of neighbors to calculate σ

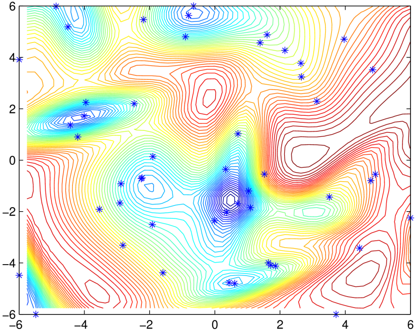
1. **If** Isampl $=1$ **Then** { *Sample N starting points* }
   else $X \leftarrow \mathrm{Uniform}(N, xl, xu)$
   Else
   else $X \leftarrow \mathrm{Halton}(N, xl, xu)$
   End If
2. **If** Ired $=1$ **Then** { *Concentrate sample points* }
   For i=1 **to** N
   $X(i) \leftarrow \mathrm{Local}(X(i), iter)$
   End For
   Else
   For i=1 **to** N
   For k=1 **to** *iter*
   $X(i) \leftarrow X(i) - \alpha \nabla f(X(i))$
   End For
   End For
   End If
3. $A \leftarrow \mathrm{Affinity}(X, Iuseg, Inei)$
   $[e_1, \ldots, e_n] \leftarrow \mathrm{Eigenvalues}(A)$
   Sort $[e_1, \ldots, e_n]$ in decreasing order and calculate maximum eigengap at k
   If Iclust $=1$ **Then** { *Apply global k-means/medoids* }
   $[M, Dis] \leftarrow \mathrm{Gkmeans}(X, k)$
   Else
   $M \leftarrow \mathrm{Gkmedoids}(A, k)$
   End If

Illustrative example



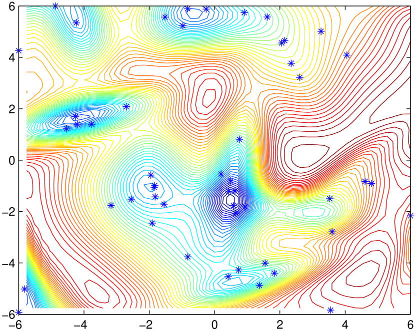
(a) iter 1

Illustrative example



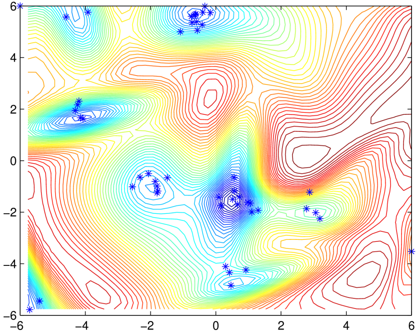
(a) iter 2

Illustrative example



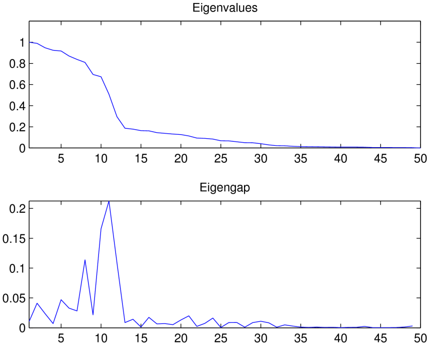
(a) iter 3

Illustrative example



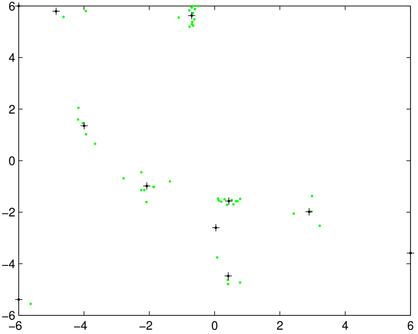
(a) iter 4

Illustrative example



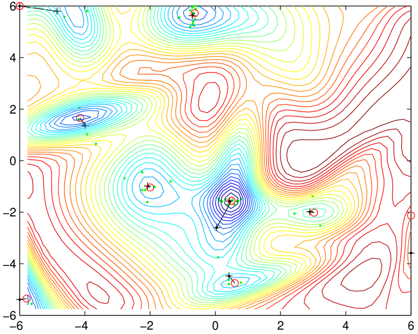
(a) Eigenvalues

Illustrative example



(a) Clustering

Illustrative example



(a) Minima retrieved

Strictly descent local search

Applicable in both frameworks!

Algorithmic framework 1

- S1. Sampling search space: Uniform or pseudo-uniform distribution
- S2. Cluster analysis: Group sampled points and assign them to minima.
- S3. Local search: Apply a local search from a representative point of each cluster.
- S4. Stopping rule: Decide whether to stop or continue.

Algorithmic framework 2

- S1. Sample from adaptive distribution: Of Implicit or explicit form.
- S2. Apply local search: Same as previous framework
- S3. Update distribution parameters: From the minima retrieved so far.
- S4. Stopping rule: Decide whether to stop or continue.

Local search goal

From a starting point create a descent sequence of iterates and converge to a minimum

Strictly descent local search

Applicable in both frameworks!

Algorithmic framework 1

- S1. Sampling search space: Uniform or pseudo-uniform distribution
- S2. Cluster analysis: Group sampled points and assign them to minima.
- S3. **Local search: Apply a local search from a representative point of each cluster.**
- S4. Stopping rule: Decide whether to stop or continue.

Algorithmic framework 2

- S1. Sample from adaptive distribution: Of Implicit or explicit form.
- S2. **Apply local search: Same as previous framework**
- S3. Update distribution parameters: From the minima retrieved so far.
- S4. Stopping rule: Decide whether to stop or continue.

Local search goal

From a starting point create a descent sequence of iterates and converge to a minimum

Early references

Kan and Timmer for theoretical analysis of their clustering algorithm:

Local search: Line search step

$x_{k+1} = x_k + a_k p_k$, p_k a descent direction and a_k a positive step

Moreover for strictly descent:

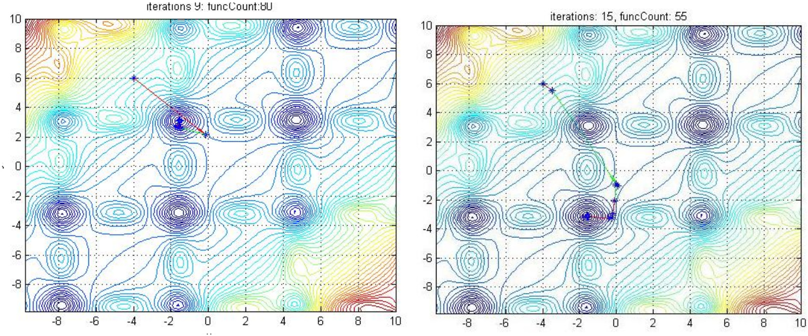
- $f(x_k + \beta p_k) \leq f(x_k + \alpha p_k)$ where $\alpha < \beta \leq a_k$
- $f(x_k + i\epsilon p_k) \leq f(x_k + (i-1)\epsilon p_k) \quad (i = 1, 2, \ldots, \left[\frac{a_k}{\epsilon}\right])$

Intuitively, the local search must define a path that is always descent

Contribution of this thesis

Although mentioned for first time before 30 years there is no practical implementation of a strictly descent local search

Example of an strictly descent local search

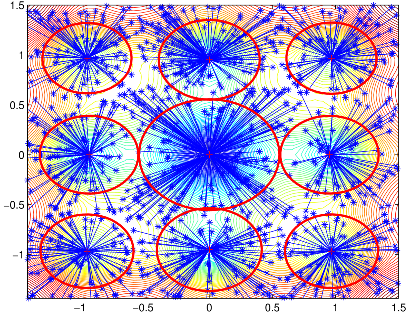


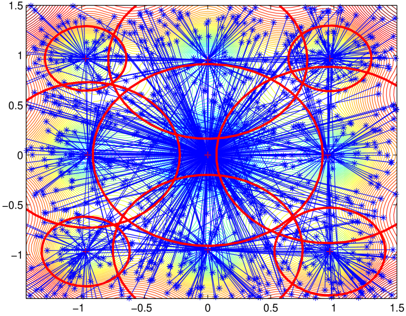
(a) Strictly local search

(b) Local search

The necessity of a new local search

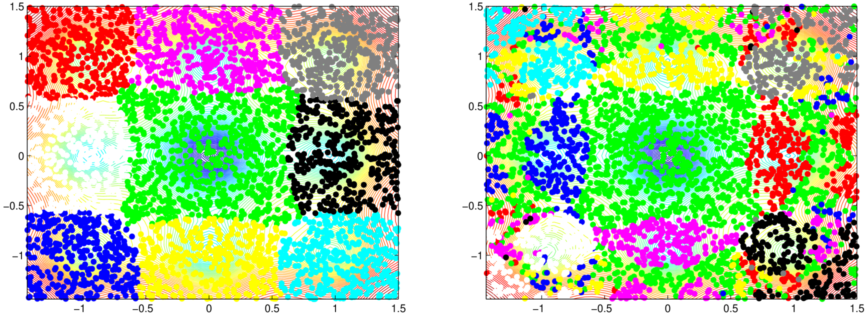
- When distance from minimum plays important role
- When modelling the region of attraction → contiguous → one model per minimum





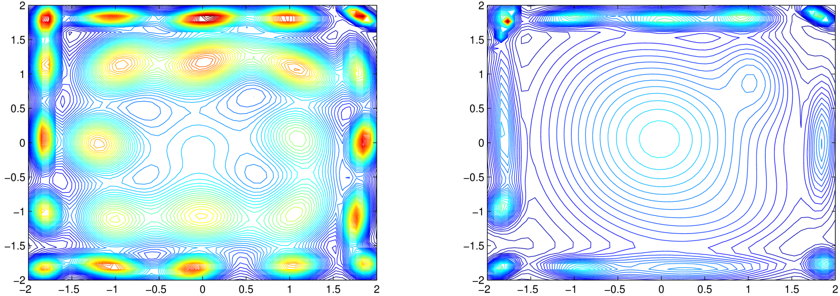
The necessity of a new local search

- When distance from minimum plays important role
- When modelling the region of attraction $\rightarrow$ contiguous $\rightarrow$ one model per minimum



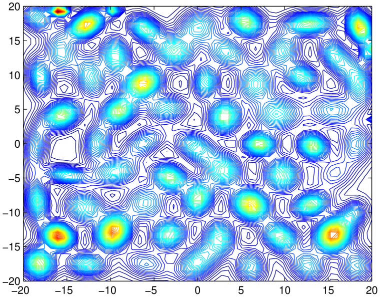
The necessity of a new local search

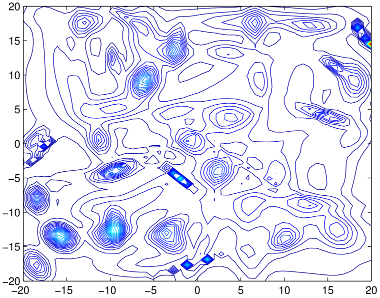
Consider the sum of normals distribution model:



The necessity of a new local search

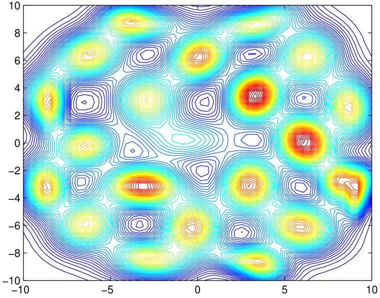
Consider the sum of normals distribution model:

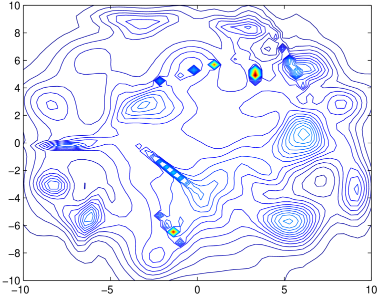




The necessity of a new local search

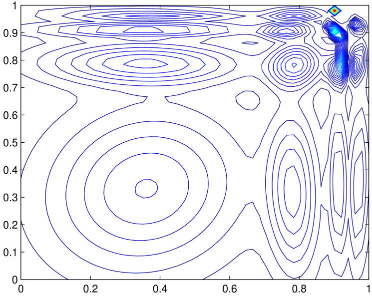
Consider the sum of normals distribution model:

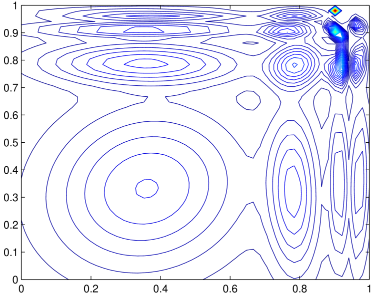




The necessity of a new local search

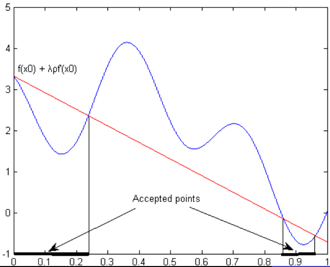
Consider the sum of normals distribution model:

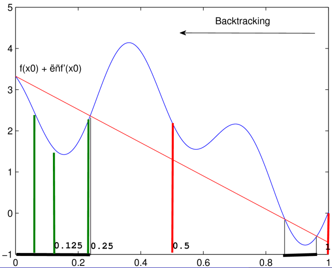




New local search properties

- Line search framework (Newton, quasi-Newton, conjugate gradient etc. provide descent direction p_k)
- Modification on a well known line search (backtracking with Armijo condition $f(x_0 + \lambda_k p_p) < f(x_0) + \lambda_k \rho f'(x_0)$)
- Ideally: The properties of ϵ-descent with the efficiency of a classical line search
- **Forward search, shortest steps near x_0**





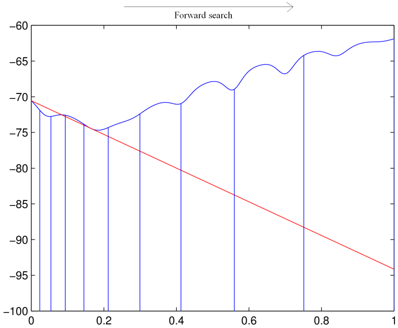
New local search

Create a grid on the permissible values:

$$\lambda_i = \frac{\mu^i - 1}{\mu^\nu - 1} \min(1, \frac{\max(1, ||x_k||)}{||s_k||})$$

Accept the first point s.t:

$$f(x_k + \lambda p_k) < f(x_k) + \rho \lambda_i p_k^T \nabla f(x_k)$$

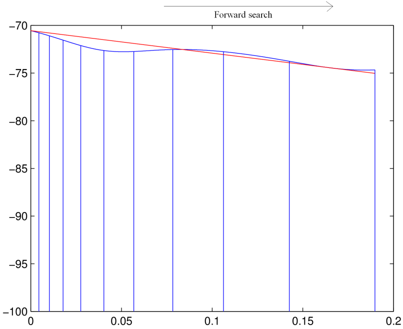


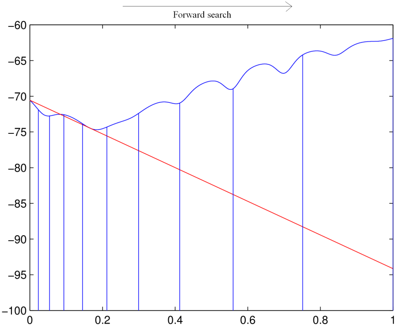
Scaling factor

Factor

$$\min(1, \frac{\max(1, ||x_k||)}{||s_k||})$$

adjusts the search to the problem's typical size.



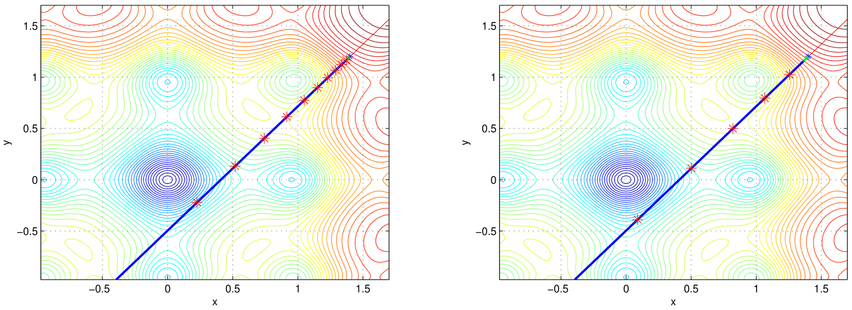


Scaling factor

Factor

$$\min(1, \frac{\max(1, ||x_k||)}{||s_k||})$$

adjusts the search to the problem's typical size.



Algorithmic description: first version

1. **Initialize:**

   $scale \leftarrow 1,\ fc \leftarrow 0,\ \mathrm{term} \leftarrow$ **false**

2. **Main Step:**

while term=**true do**
    for i=1, ν **do**

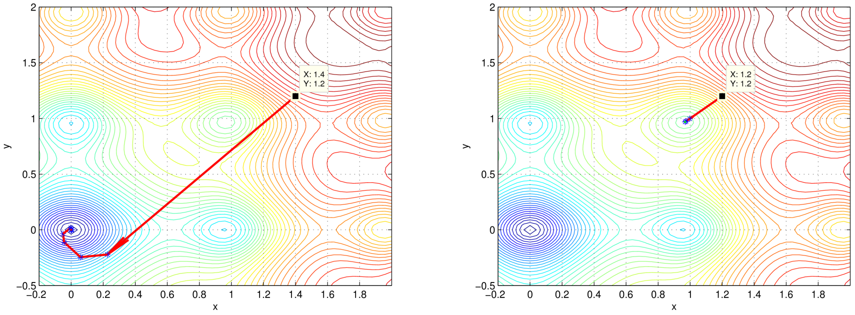
$$\lambda_i \leftarrow scale \cdot \frac{\mu^i - 1}{\mu^\nu - 1} \cdot \min\left(1, \frac{\max(1, ||x||)}{||p_k||}\right)$$

        if $f(x + \lambda_i p_k) < f(x) + \rho \lambda_i \cdot p_k^T \nabla f(x)$ **then** { *Bellow* ρ *line* }
            if $f(x + \lambda_i p_k) > f(x + \lambda_{i-1} p_k)$ **then** { *No improvement* }
                $\alpha \leftarrow \lambda_{i-1}$
                $x' \leftarrow x + \alpha p_k$
                term $\leftarrow$ **true**, **break**
            end if
        else { *Above* ρ *line* }
            $\alpha \leftarrow \lambda_{i-1}$
            $x' \leftarrow x + \alpha p_k$
            term $\leftarrow$ **true**, **break**
        end if
        $fc \leftarrow fc + 1$
    end

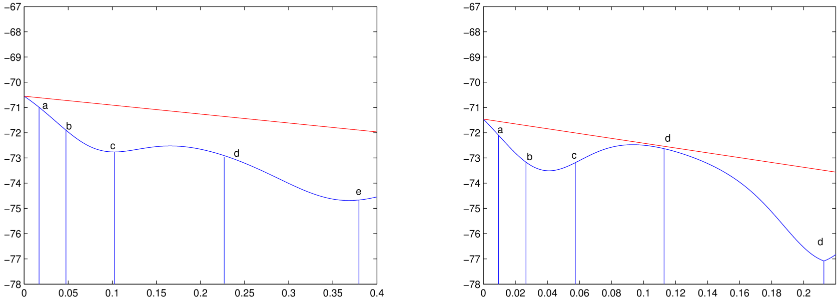
$$scale \leftarrow scale \frac{\mu^i - 1}{\mu^\nu - 1} \cdot \min\left(1, \frac{\max(1, ||x||)}{||p_k||}\right)$$

end

First version need further information

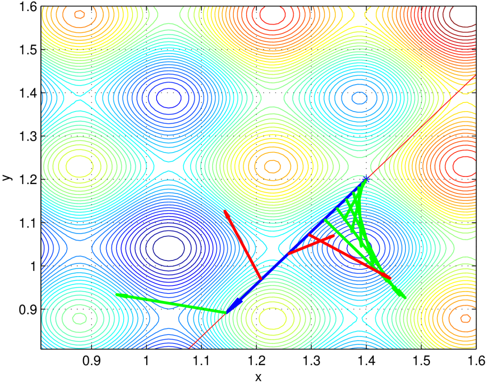


First version need further information



Adding derivative information

Idea: Dot product of the direction p_k and $\nabla f(x_0 + \lambda_i p_k)$ changes sign.



Corrects previous case

Algorithmic description: adding gradient information

1. **Main Step:**

while term=**true** **do**
  for i=1, ν **do**
    $\lambda_i \leftarrow scale \cdot \frac{\mu^i - 1}{\mu^\nu - 1} \cdot \min\left(1, \frac{\max(1, ||x||)}{||p_k||}\right)$
    if $f(x + \lambda_i p_k) < f(x) + \rho\lambda_i \cdot p_k^T \nabla f(x)$ **then** { *Bellow ρ line* }
      if $f(x + \lambda_i p_k) > f(x + \lambda_{i-1} p_k)$ **then** { *No improvement* }
        $\alpha \leftarrow \lambda_{i-1}$
        $x' \leftarrow x + \alpha p_k$
        term $\leftarrow$ **true**, **break**
      else { *Bellow ρ line and improving* }
        $g_i \leftarrow \nabla f(x_i)$
        if $g_i^T p_k > 0$ **then**
          $\alpha \leftarrow \lambda_{i-1}$
          $x' \leftarrow x + \alpha p_k$
          term $\leftarrow$ **true**, **break**
        end if
      end if
    else { *Above ρ line* }
      $\alpha \leftarrow \lambda_{i-1}$
      $x' \leftarrow x + \alpha p_k$
      term $\leftarrow$ **true**, **break**
    end if
  end
  $scale \leftarrow scale \frac{\mu^i - 1}{\mu^\nu - 1} \cdot \min\left(1, \frac{\max(1, ||x||)}{||p_k||}\right)$
end

Accelerate: choosing ν

Observation

Line search algorithms take full steps near the minimum.

$$x_{k+1} = x_k + \lambda p_{k+1}, \quad \lambda = 1$$

Consequence: The proposed line search $\rightarrow \nu$ function calls

Solution

Estimate parameter ν adaptively, based on quantity $h = \dfrac{1}{1 - p_k^T \nabla f(x_k)}$

Define the first step on the grid to be h, from this estimate ν'

$h \rightarrow 1 \Rightarrow \nu' \rightarrow 1$ when $p_k^T \nabla f(x_k) \rightarrow 0$ (near minimum)

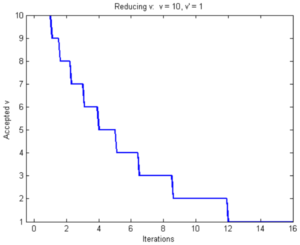
Accept $\nu = \max\left(\left\lfloor (\nu' - \nu) \cdot e^{0.1(-iter+1)} + \nu \right\rfloor, 1\right)$

Algorithmic description: choosing ν

1. **Initialize:**

$$scale \leftarrow 1,\ \ fc \leftarrow 0,\ \ gc \leftarrow 0$$
$$h \leftarrow \frac{1}{1-p_k^T \nabla f(x)}$$
$$sc \leftarrow \min\left(1, \max\left(1, \frac{||x||}{||\nabla f(x)||}\right)\right)$$
$$\nu' \leftarrow \nu \quad , \nu \leftarrow \left\lfloor \min\left(\nu', \frac{\log(1+(\mu-1)\frac{sc}{h})}{\log(\mu)}\right)\right\rfloor$$
$$\nu = \max\left(\left\lfloor(\nu' - \nu)\cdot e^{0.1(-iter+1)} + \nu\right\rfloor, 1\right)$$

2. **Main Step:**



Accuracy vs. Efficiency

Armijo Type Local Search							Local Local Search n=10, m=1.3					
Function	Correct	Error	Iters.	Fun. Calls	% Success		Function	Correct	Error	Iters.	Fun. Calls	%
Ackley	555	445	15704	24978	55,50%		Ackley	874	126	11555	61583	87,40%
Giunta	588	442	12013	15855	57,09%		Giunta	912	95	12891	83760	90,57%
Guillin	477	523	17764	29974	47,70%		Guillin	896	109	10211	55838	89,15%
Levy3	1285	715	20783	31624	64,25%		Levy3	1910	90	19882	139418	95,50%
Rastrigin	599	401	9240	12822	59,90%		Rastrigin	1000	0	9031	58571	100,00%
Griewank	695	305	10244	13422	69,50%		Griewank	998	2	9403	72245	99,80%
Bird	704	296	14104	19960	70,40%		Bird	913	87	10782	78446	91,30%
Levy5	1272	728	21476	31377	63,60%		Levy5	1811	189	19282	21956	90,55%
Rot. Quad	539	461	12001	18408	53,90%		Rot. Quad	903	97	9964	69611	90,30%
Holder	597	403	12235	17430	59,70%		Holder	995	5	10554	63490	99,50%
Liang	561	439	11671	18912	56,10%		Liang	702	298	11782	87584	70,20%
Piccioni	726	274	24874	44702	72,60%		Piccioni	997	3	9021	81703	99,70%
Shekel	145	155	3757	7493	48,33%		Shekel	250	50	3783	24281	83,33%
M0	717	1283	132739	142455	35,85%		M0	1200	800	19826	132526	60,00%
Lager	581	419	11342	16537	58,10%		Lager	901	99	10435	69524	90,10%
Tube	727	273	10034	13719	72,70%		Tube	1000	0	8991	50724	100,00%
Mich	178	322	11472	16222	35,60%		Mich	356	144	11282	55817	71,20%
Dejong	301	199	18417	22221	60,20%		Dejong	489	12	16822	53632	97,60%
Sum / Ave.	11247	8083	369870	498111	57.8%		Sum / Ave	17107	2206	215497	1260710,6	89,23%

A new stopping rule

Applicable in both frameworks!

Algorithmic framework 1

- S1. Sampling search space: Uniform or pseudo-uniform distribution
- S2. Cluster analysis: Group sampled points and assign them to minima.
- S3. Local search: Apply a local search from a representative point of each cluster.
- S4. Stopping rule: Decide whether to stop or continue.

Algorithmic framework 2

- S1. Sample from adaptive distribution: Of Implicit or explicit form.
- S2. Apply local search: Same as previous framework
- S3. Update distribution parameters: From the minima retrieved so far.
- S4. Stopping rule: Decide whether to stop or continue.

A new stopping rule

Applicable in both frameworks!

Algorithmic framework 1

S1. Sampling search space: Uniform or pseudo-uniform distribution

S2. Cluster analysis: Group sampled points and assign them to minima.

S3. Local search: Apply a local search from a representative point of each cluster.

S4. **Stopping rule: Decide whether to stop or continue.**

Algorithmic framework 2

S1. Sample from adaptive distribution: Of Implicit or explicit form.

S2. Apply local search: Same as previous framework

S3. Update distribution parameters: From the minima retrieved so far.

S4. **Stopping rule: Decide whether to stop or continue.**

Stopping rules requirements

Trade off between efficiency (speed) and quality (global minima).
Requirements

- **Sample dependent**: The actual objective function values and their location, or the number of times that local optima are identified by a local search procedure.
- **Problem dependent**: Maximal use should be made of available prior information. This information may concern, for instance, the number of local optima and the size of the regions of attraction, or the tail of the distribution of function values.
- **Method dependent**: If some general algorithmic properties of the applied method are known, these should be incorporated in the stopping rule.
- **Loss dependent**: Stopping rules should take into account the seriousness of the cost incurred if the search is terminated before the global optimum is identified.
- **Resource dependent**: Evidently the computational effort should be kept as small as possible.

Taxonomy of stopping rules

- Naive thresholding
- Stopping rules based on the local optima structure
  - Statistical models
  - Non-sequential rules (**Zielinski, Boender et al**)
  - Sequential rules
  - Incorporation of function values
- Stopping rules based on coverage of search space (**Double Box**)
- Stopping rules based on the distribution of function values
  - Continuous case
  - Discrete case

Stopping rule idea

Problem definition

Consider a box containing w different balls. The balls are numbered sequentially $1, 2, 3, \ldots, w$. We pick a ball at random examine its number, and we put it back in the box. This is one iteration. If the ball number has not been drawn previously we update the distinct ball count m, otherwise we don't.

Stopping rule idea

Problem definition

Consider a box containing w different balls. The balls are numbered sequentially $1, 2, 3, \ldots, w$. We pick a ball at random examine its number, and we put it back in the box. This is one iteration. If the ball number has not been drawn previously we update the distinct ball count m, otherwise we don't.

Correspondence to optimization

We pick a ball at random examine its number $\longleftrightarrow$ application of local optimization

Stopping rule idea

Problem definition

Consider a box containing w different balls. The balls are numbered sequentially $1, 2, 3, \ldots, w$. We pick a ball at random examine its number, and we put it back in the box. This is one iteration. If the ball number has not been drawn previously we update the distinct ball count m, otherwise we don't.

Correspondence to optimization

We pick a ball at random examine its number $\longleftrightarrow$ application of local optimization

Stopping rule idea

Problem definition

Consider a box containing w different balls. The balls are numbered sequentially $1, 2, 3, \ldots, w$. We pick a ball at random examine its number, and we put it back in the box. This is one iteration. If the ball number has not been drawn previously we update the distinct ball count m, otherwise we don't.

Correspondence to optimization

We pick a ball at random examine its number $\longleftrightarrow$ **application of local optimization**

Basic formulae I

At iteration k, the probability that m balls (minima) are found is denoted by $p_m^{(k)}$

Expected number of *distinct* balls after k iterations

$$< N >^{(k)} = \sum_{i=1}^{k} i \cdot p_i^{(k)} = p_1^{(k)} + 2p_2^{(k)} + \cdots + kp_k^{(k)}$$

Recursive definition of $p_m^{(k)}$

$$p_i^{(k+1)} = \alpha p_i^{(k)} + \beta p_{i-1}^{(k)}$$

- the probability that in the previous iteration ($k-$th), i minima were already recovered and in the $(k+1)-$th no new minimum is found (this is with probability α),
- the probability that in the $k-$th iteration $(i-1)$ minima were found and in the $(k+1)-$th iteration one more minimum (new) is found (with probability β).

Basic formulae II

The task of calculating $p_i^{(k)}$ is now reduced to the task of defining the probabilities α and β.
Assumption:

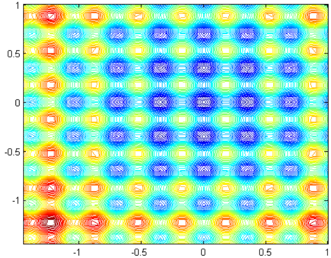
> *The probability of locating a local minimum, among the w distinct ones, by applying a local search is $p = \frac{1}{w}$.*
>
> *All minima are retrieved (by applying a local search) with uniform probability.*

Rationalize the assumption

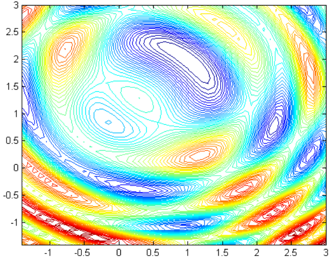
The above assumption although unachieved in the multistart framework, it makes sense in the concept of stochastic clustering algorithms were we (optimally) aim to perform *one local search per minimum*.

- $\alpha = \frac{i}{w}$
- $\beta = \frac{w-(i-1)}{w}$

Example test functions



(a) Uniform probability



(b) Non-uniform probability

Stopping rule made practical

Let $N_{found}^{(i)}$ the number of distinct minima at i-th iteration. Then the quantity:

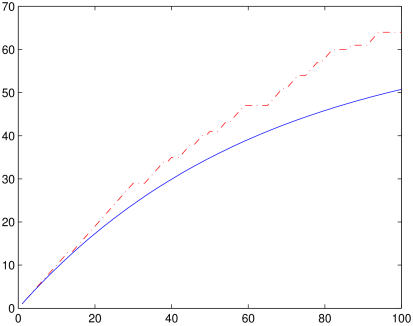
$$d_{MSE} = \frac{1}{iter} \sum_{i=0}^{iter} \left(< N^{(i)} > - N_{found}^{(i)} \right)^2$$

is decreasing to zero.

Iteration	Minima Found	MSE	Variance
100	64	78.949828	205.044716
200	94	72.086818	3 6.688531
300	110	52.139962	9.248131
400	116	28.281410	2.063600
500	119	10.676961	0.443810
600	121	9.770836	0.095597
700	121	5.789973	0.018181
800	121	2.525016	0.003458
900	121	1.101163	0.000658

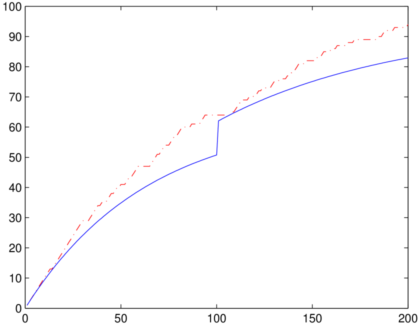
Table: Expected number of minima vs. the real minima found

Illustration



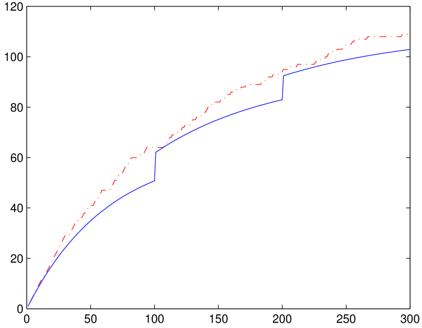
(c) Iter 100

Illustration



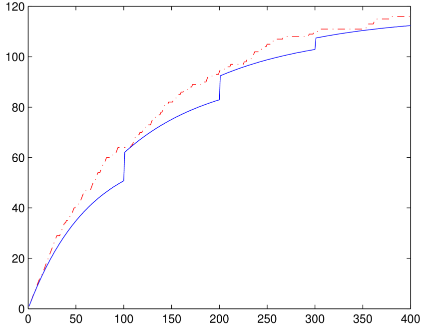
(d) Iter 200

Illustration



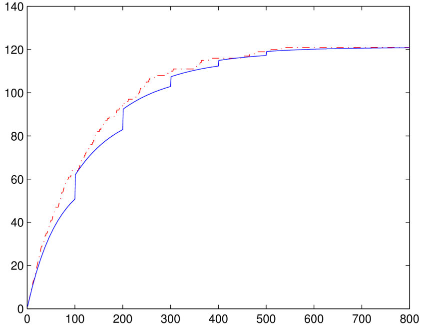
(e) Iter 300

Illustration



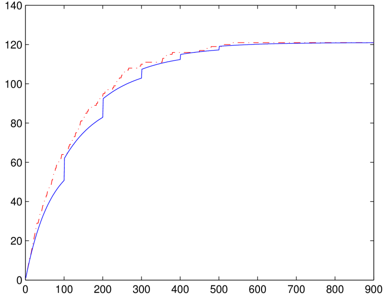
(f) Iter 400

Illustration



(g) Iter 800

Illustration



(h) Iter 900

Description of compared stopping criteria

- **Zielinski**: Fraction of uncovered space: $P(w) = \frac{w(w+1)}{t(t-1)}$, Stop when $P(w) \leq \epsilon$
- **Boender et al**: Estimated number of local minima $w_{\text{est}} = \frac{w(t-1)}{t-w-2}$, Stop when $w_{\text{est}} - w \leq \frac{1}{2}$
- **Double box(Tsoulos&Lagaris)**: Suppose sampling from a larger space. $\delta_k \equiv \frac{k}{M_k}$, Stop when $\sigma_k^2(\delta) \leq \sigma_k^2(\delta_{last})$
- **Proposed criterion**: $d_{MSE} = \frac{1}{k} \sum_{i=0}^{k} \left(< N^{(i)} > - N_{found}^{(i)} \right)^2$, Stop when $\sigma_k^2(d_{MSE}) < \epsilon$

Experimental comparison

Function	Zielinski			Rinnoy-Kan			Tsoulos-Lagaris			Proposed		
	nom	nloc	feval	nom	nloc	feval	nom	nloc	feval	nom	nloc	feval
Rast(121)	121	3843	66863	121	14886	254412	121	2129	36903	121	1500	25905
Ack(49)	49	1566	42498	49	2502	67686	48.6	1079	29081	48.6	615	16457
Gri(123)	123	3906	66436	123	15378	261801	123	1842	31414	123	1500	25742
Lev3(130)	130	4128	79330	130	17163	329956	130	2078	40043	130	1605	30877
Lev5(130)	130	4128	92642	130	17163	383216	130	2206	49075	130	1605	35718
Lag(64)	63.8	2035	39807	64	4227	82401	63.95	2859	55803	62.8	845	16607
R-G(94)	92.65	2947	78023	93.2	8875.8	235069	92.6	4503	119098	91.65	1185	31340
Giu(36)	36	1155	17007	36	1371	20208	35.95	432	6405	36	500	7324
Gui	303.45	9612	190830	369.9	20000	396869	371	20000	396951	369.7	20000	396951
M0(152)	151.45	4806	71885	154.45	20000	299674	152.55	11265	168760	152	1920	28989
M5(441)	440.9	13959	1215763	440.85	20000	1742052	440.85	18623	1621518	439	5800	505057
She(10)	10	333	10183	10	123	3776	10	158.2	4808,25	10	200[a]	6064
Bir(25)	24.95	805	22764	24.8	668	19019	22.8	659	18639	24.85	420	12025
Tub(45)	45	1440	18457	45	2118	27140	45	483	6174	45	600	7863
Dej(64)	62.65	1997	103142	63	4099	211941	62.2	1134	58701	63	800	41508
Hol(180)	180	5709	111733	180	20000	391459	180	2881	56357	180	2300	45052
Pic(37)	37	1187	25792	37	1446	31472	37	1036	22457	37	520	10997

[a] The minimum number of local searched needed to start evaluate our stopping rule

Part II

Local Optimization

Presentation Outline

9. Convex Quadratic Programming With Bound Constraints
10. A Rectangular Trust Region Optimization Algorithm
11. A Hybrid Local Search For Neural Network Training

Problem Definition

The Quadratic Programming problem with simple bounds is stated as:

$$q(x) = \min_x \ \frac{1}{2} x^T B x + x^T d, \tag{1}$$

$$\text{subject to: } a_i \leq x_i \leq b_i, \forall i \in I = \{1, 2, \cdots, n\}$$

where $x, d \in R^n$ and B is a symmetric, positive definite $n \times n$ matrix.

Applications of Quadratic Programming with simple bounds

- Computational Physics
- Engineering
- Training Support Vector Machines
- Biomedical Applications (Radiation Intensity Optimization)
- Part of Optimization Algorithm (see Rectangular Trust Region)

Solution Methods

- Active Set Techniques: Iterates on a face of the feasible box until either a minimizer of the objective function is found or a point on the boundary of that face is reached.
- Gradient Projection: Like active set, but allowing more than one faces of the feasible box
- Interior Point Techniques: In brief, an interior point algorithm consists of adding a series of parameterized barrier functions which are minimized using Newton's method

KKT Conditions

Quadratic Problem

$$q(x) = \min_{x} \ \frac{1}{2} x^T B x + x^T d,$$
$$\text{subject to: } a_i \leq x_i \leq b_i, \forall i \in I = \{1, 2, \cdots, n\}$$

Lagrangian

$$L(x, \lambda, \mu) = \frac{1}{2} x^T B x + x^T d - \lambda^T (x - a) - \mu^T (b - x)$$

KKT Conditions for Quadratic Problem

$$Bx^* + d - \lambda^* + \mu^* = 0 \tag{2}$$
$$\lambda_i^* \geq 0, \quad \mu_i^* \geq 0, \quad \forall i \in I \tag{3}$$
$$\lambda_i^* (x_i^* - a_i) = 0, \quad \forall i \in I \tag{4}$$
$$\mu_i^* (b_i - x_i^*) = 0, \quad \forall i \in I \tag{5}$$
$$x_i^* \in [a_i, b_i], \quad \forall i \in I \tag{6}$$

Active Set A

$$A = \{ i \in I : \mu_i^* \geq 0 \text{ or } \lambda_i^* \geq 0 \}$$

Basic sketch

- Given the Active Set A (let $\tilde{S} = I - A$), solve the system

$$B_{\tilde{S}\tilde{S}} x_{\tilde{S}} = -d_{\tilde{S}}$$

- The first KKT condition is a *nxn* linear system but the number of unknowns is $3n$ (x, λ, μ)
- Construct three sets $L^{(k)}, U^{(k)}, S^{(k)}$ such that $I = L^{(k)} \cup U^{(k)} \cup S^{(k)}$ and $L^{(k)} \cap S^{(k)} = L^{(k)} \cap U^{(k)} = U^{(k)} \cap S^{(k)} = \emptyset$
- Start with a guess for the solution $(x^{(0)})$
- Enforce the complementarity conditions (3) and (4) of KKT Conditions to obtain $\lambda^{(0)}$ and $\mu^{(0)}$.
- Solve a reduce linear system for the next iteration $(x^{(1)})$.
- Calculate using substitution from Eq.(1) of KKT $\lambda^{(1)}$ and $\mu^{(1)}$

The Algorithm (I)

Algorithm BOXCQP

Initially set: $k = 0$, $\lambda^{(0)} = \mu^{(0)} = 0$ and $x^{(0)} = -B^{-1}d$.
If $x^{(0)}$ is feasible, **Stop**, the solution is: $x^* = x^{(0)}$.
At iteration k, the quantities $x^{(k)}, \lambda^{(k)}, \mu^{(k)}$ are available.

1. Define the sets:

$$\begin{aligned} L^{(k)} &= \{i : x_i^{(k)} < a_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} \geq 0\} \\ U^{(k)} &= \{i : x_i^{(k)} > b_i, \text{ or } x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} \geq 0\} \\ S^{(k)} &= \{i : a_i < x_i^{(k)} < b_i, \text{ or } x_i^{(k)} = a_i \text{ and } \lambda_i^{(k)} < 0, \\ \text{or} & \quad x_i^{(k)} = b_i \text{ and } \mu_i^{(k)} < 0\} \end{aligned}$$

   Note that $L^{(k)} \cup U^{(k)} \cup S^{(k)} = I$

2. Set:

$$\begin{aligned} x_i^{(k+1)} &= a_i, \; \mu_i^{(k+1)} = 0, \quad \forall i \in L^{(k)} \\ x_i^{(k+1)} &= b_i, \; \lambda_i^{(k+1)} = 0, \quad \forall i \in U^{(k)} \\ \lambda_i^{(k+1)} &= 0, \; \mu_i^{(k+1)} = 0, \quad \forall i \in S^{(k)} \end{aligned}$$

The Algorithm (II)

3. Solve:

$$Bx^{(k+1)} + d = \lambda^{(k+1)} - \mu^{(k+1)}$$

for the n unknowns:

$$x_i^{(k+1)},\ \forall i\ \in S^{(k)}$$
$$\mu_i^{(k+1)},\ \forall i\ \in U^{(k)}$$
$$\lambda_i^{(k+1)},\ \forall i\ \in L^{(k)}$$

4. Check if the new point is a solution and decide to either stop or iterate.

If $(x_i^{(k+1)} \in [a_i, b_i]\ \forall i \in S^{(k)}$ **and** $\mu_i^{(k+1)} \geq 0,\ \forall i \in U^{(k)}$
and $\lambda_i^{(k+1)} \geq 0,\ \forall i \in L^{(k)})$ **Then**
Stop, the solution is: $x^* = x^{(k+1)}$.
Else
set $k \leftarrow k+1$ and iterate from **Step 1.**
Endif

Solution of Linear System (Step 3)

Linear System Step 3

$$\sum_{j\in I} B_{ij}x_j^{(k+1)} + d_i = \lambda_i^{(k+1)} - \mu_i^{(k+1)}, \ \forall i \in I$$

$\forall i \in S^{(k)}$ we have that $\lambda_i^{(k+1)} = \mu_i^{(k+1)} = 0$, hence we can calculate $x_i^{(k+1)}, \ \forall i \in S^{(k)}$

Splitting the sum, calculate x

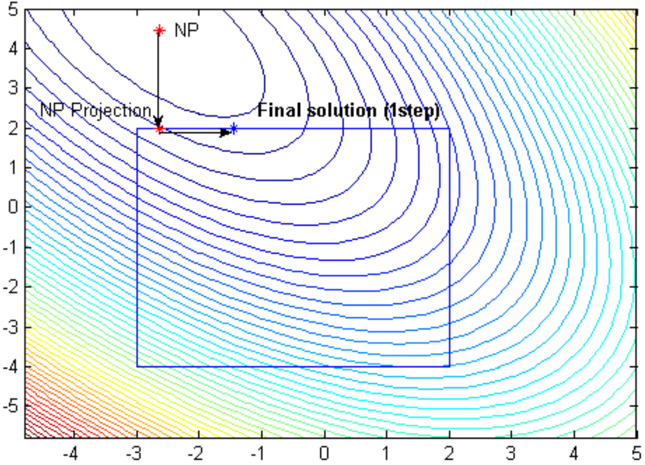
$$\sum_{j\in S^{(k)}} B_{ij}x_j^{(k+1)} = -\sum_{j\in L^{(k)}} B_{ij}a_j - \sum_{j\in U^{(k)}} B_{ij}b_j - d_i, \ \forall i \in S^{(k)}$$

Calculate λ and μ

$$\lambda_i^{(k+1)} = \sum_{j\in I} B_{ij}x_j^{(k+1)} + d_i, \ \forall i \in L^{(k)}$$

$$\mu_i^{(k+1)} = -\sum_{j\in I} B_{ij}x_j^{(k+1)} - d_i, \ \forall i \in U^{(k)}$$

Illustration



Properties of the algorithm

- Depends on the condition number of B
- For well conditioned problems only a few iterations are performed
- No convergence theory (yet!)

If $L^{(k+1)} = L^{(k)}$ and $U^{(k+1)} = U^{(k)}$ and $S^{(k+1)} = S^{(k)}$ then $L^{(k)}, U^{(k)}, S^{(k)}$ satisfy the KKT conditions.

Experimental Testbed

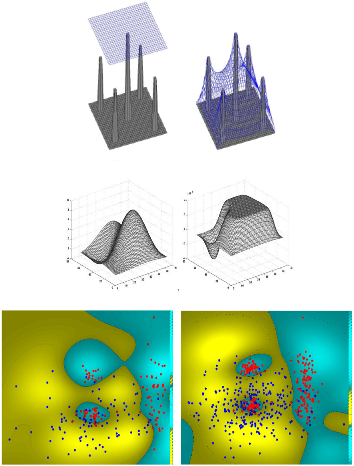
- Matlab Implementation
  - **MOSEK** Commercial optimization product that includes Convex Quadratic solver.
  - **Quadprog** Matlab's own Quadratic solver
- Fortran Implementation
  - **QPBOX** is a Fortran77 package for box constrained quadratic programs developed at IMM of the Technical University of Denmark.
  - **QLD** This program [?] is due to K.Schittkowski of the University of Bayreuth, Germany.
  - **QUANCAN** This program combines conjugate gradients and gradient projection techniques

BoxCQP Variations

- Conjugate gradient solver
- Cholesky factorization solver
- Mixed approach

Test Problems

- Random problems
  - R1: Equal probability for upper-lower
  - R2: 90% of the variables on upper bound
  - R3: 90% of the variables on lower bound
- Circus Tent
- Biharmonic Equation
- Intensity Modulation Radiation Therapy
- Support Vector Classification



Prob. Name	Var.1	Var.2	Var.3	QUACAN	QPBOX	QLD
SVM ($n=100$), $f^* = -167.79$	0.00	0.00	0.00	0.01	0.01	0.00
SVM ($n=200$), $f^* = -384.11$	0.0430	0.0352	0.0391	0.1523	0.0703	0.0742
SVM ($n=300$), $f^* = -545.54$	0.1367	0.0977	0.1016	0.3945	0.2305	0.2539
SVM ($n=400$), $f^* = -736.00$	0.3672	0.2891	0.3359	2.0039	0.5352	0.6289
SVM ($n=500$), $f^* = -933.94$	0.7031	0.6133	0.7070	4.6914	1.0508	1.2734
SVM ($n=600$), $f^* = -1073.77$	1.1797	0.8398	0.9727	6.5430	1.8516	2.3633
SVM ($n=700$), $f^* = -1222.33$	2.2656	1.5078	1.8516	14.3320	3.0273	3.8125
SVM ($n=800$), $f^* = -1323.44$	3.3789	1.8750	2.2539	21.7461	4.6836	6.1953
SVM ($n=900$), $f^* = -1431.59$	5.6680	3.3984	3.8438	27.1602	7.3281	8.2031
SVM ($n=1000$), $f^* = -1539.77$	7.2578	4.2930	5.0117	34.0078	10.3945	11.3281
SVM ($n=2000$), $f^* = -2849.68$	68.7852	22.0078	36.2461	256.2266	77.2969	104.3086
SVM ($n=3000$), $f^* = -4490.68$	263.3477	63.9688	151.4023	1068.9766	264.5586	354.4297
Tent ($n=100$), $f^* = 0.0168$	0.0078	0.0039	0.0039	0.0039	$N.C$	0.0039
Tent ($n=400$), $f^* = 0.3162$	0.322	0.132	0.217	$N.C$	$N.C$	0.248
Tent ($n=900$), $f^* = 0.4442$	5.570	1.453	3.273	$N.C$	$N.C$	2.77
Tent ($n=1600$), $f^* = 0.5023$	48.3008	9.5742	29.1133	$N.C$	$N.C$	20.5352
Tent ($n=3600$), $f^* = 0.5455$	557.74	55.74	284.05	$N.C$	$N.C$	246.04
Tent ($n=4900$), $f^* = 0.5540$	1333.51	150.49	696.21	$N.C$	$N.C$	617.58
Biharm ($n=100$), $f^* = -0.0001$	0.0030	0.0030	0.0020	0.0040	0.0120	0.0130
Biharm ($n=400$), $f^* = -0.0004$	0.1958	0.2090	0.1880	0.6450	0.6382	0.7539
Biharm ($n=900$), $f^* = -0.0008$	4.2788	3.1328	2.9180	18.5229	10.4912	9.2886
Biharm ($n=1600$), $f^* = -0.0015$	23.3280	17.8920	15.3110	119.6610	82.1220	60.8680
Biharm ($n=2500$), $f^* = -0.0023$	106.1869	77.2411	60.5740	775.0340	333.9110	222.7870
Biharm ($n=3600$), $f^* = -0.0033$	308.7246	271.4639	186.8857	2988.0826	1071.3447	684.5688
Biharm ($n=4900$), $f^* = -0.0045$	816.13	705.52	484.45	8282.04	3067.21	1837.66
IMRT ($n=2342$), $f^* = 0.0563$	54.22	33.11	40.56	85.11	67.88	73.22

Prob. Name	Var.1	Var.2	Var.3	QUACAN	QPBOX	QLD
Random ($ncond = 0.1$, $n = 500$, $f^* = -762.53$)	0.40	0.05	0.09	0.10	0.86	1.23
Random ($ncond = 1$, $n = 500$, $f^* = -1133.68$)	0.4141	0.15	0.18	0.48	0.85	1.19
Random ($ncond = 5$, $n = 500$, $f^* = -1692549$)	0.63	4.04	0.87	48.75	1.24	1.25
Random ($ncond = 0.1$, $n = 600$, $f^* = -994.19$)	0.78	0.06	0.12	0.14	1.43	2.13
Random ($ncond = 1$, $n = 600$, $f^* = -1288.29$)	0.90	0.24	0.37	0.66	1.57	2.14
Random ($ncond = 5$, $n = 600$, $f^* = -2049820$)	1.16	9.72	1.27	48.69	1.97	2.14
Random ($ncond = 0.1$, $n = 700$, $f^* = -838.28$)	1.34	0.09	0.20	0.23	2.35	3.46
Random ($ncond = 1$, $n = 700$, $f^* = -1703.28$)	1.31	0.28	0.34	0.73	2.45	3.68
Random ($ncond = 5$, $n = 700$, $f^* = -2328669$)	2.49	20.45	2.84	164.35	3.92	3.45
Random ($ncond = 0.1$, $n = 800$, $f^* = -645.14$)	2.58	0.13	0.27	0.31	3.80	5.44
Random ($ncond = 1$, $n = 800$, $f^* = -1824.65$)	2.59	0.42	0.53	1.26	3.76	5.37
Random ($ncond = 5$, $n = 800$, $f^* = -2630417$)	3.58	32.21	3.72	108.13	5.76	5.47
Random ($ncond = 0.1$, $n = 900$, $f^* = -596.17$)	4.02	0.19	0.40	0.68	5.70	7.50
Random ($ncond = 1$, $n = 900$, $f^* = -1951.62$)	4.04	0.63	0.77	2.13	5.60	7.44
Random ($ncond = 5$, $n = 900$, $f^* = -2904251$)	5.16	46.01	5.87	145.91	7.39	7.64
Random ($ncond = 0.1$, $n = 1000$, $f^* = -1327.91$)	4.52	0.22	0.54	0.50	7.83	10.17
Random ($ncond = 1$, $n = 1000$, $f^* = -2677.47$)	4.58	0.66	0.95	1.68	7.93	10.00
Random ($ncond = 5$, $n = 1000$, $f^* = -3082720$)	6.82	72.48	8.19	143.93	10.02	9.93
Random ($ncond = 0.1$, $n = 1100$, $f^* = -1464.97$)	7.86	0.35	0.77	0.81	10.69	13.79
Random ($ncond = 1$, $n = 1100$, $f^* = -2061.31$)	7.98	1.01	1.45	3.50	10.31	13.57
Random ($ncond = 5$, $n = 1100$, $f^* = -4224564$)	10.84	85.98	12.03	213.99	14.21	13.77
Random ($ncond = 0.1$, $n = 1200$, $f^* = -1332.93$)	9.40	0.33	1.23	0.75	13.26	18.05
Random ($ncond = 1$, $n = 1200$, $f^* = -1978.65$)	9.02	0.96	2.29	3.32	13.49	19.19
Random ($ncond = 5$, $n = 1200$, $f^* = -4071507$)	11.08	90.54	11.95	187.50	16.90	19.31
Random ($ncond = 0.1$, $n = 1300$, $f^* = -2247.07$)	10.73	0.41	1.29	1.14	17.33	23.89
Random ($ncond = 1$, $n = 1300$, $f^* = -2698.07$)	11.67	1.46	2.91	4.43	17.73	24.35
Random ($ncond = 5$, $n = 1400$, $f^* = -1537.81$)	16.66	136.48	20.56	733.69	25.75	24.06
Random ($ncond = 0.1$, $n = 1400$, $f^* = -2247.07$)	12.03	0.43	1.57	1.14	20.94	30.16
Random ($ncond = 1$, $n = 1400$, $f^* = -2860.81$)	11.97	1.20	2.15	3.51	21.41	30.57
Random ($ncond = 5$, $n = 1400$, $f^* = -4446068$)	17.48	118.56	17.92	300.58	27.72	30.07
Random ($ncond = 0.1$, $n = 1500$, $f^* = -1287.82$)	17.70	0.50	2.16	1.14	25.30	36.80
Random ($ncond = 1$, $n = 1500$, $f^* = -2952.20$)	20.42	1.92	5.78	5.21	26.37	35.48
Random ($ncond = 5$, $n = 1500$, $f^* = -3811836$)	27.47	226.89	34.67	624.51	47.77	36.22

Trust Region Idea

Problems applied

$$\min_{x \in \mathcal{R}^n} f(x) \quad \text{subject to } a_i \leq x \leq b_i$$

Approximate $f(x)$

$$f(x_k + s) \approx m_k(s) = f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s \tag{7}$$

where $g_k = \nabla f(x_k)$ and B_k is a symmetric approximation to $\nabla^2 f(x_k)$.
The trust region may be defined by:

$$\mathbf{T}_k = \{x \in \Re^n \mid ||x - x_k|| \leq \Delta_k\} \tag{8}$$

It is obvious that different choices for the norm lead to different trust region shapes. The Euclidean norm $|| \cdot ||_2$, corresponds to a hypershpere, while the $|| \cdot ||_\infty$ norm defines a hyperbox.

Trust Region Basic Algorithm

Basic trust region

S0: **Pick the initial point and trust region parameter x_0 and Δ_0, and set $k = 0$.**

S1: Construct a quadratic model:
$m_k(s) \approx f(x_k + s)$

S2: Calculate s_k with $||s_k|| \leq \Delta_k$, so as to sufficiently reduce m_k.

S3: Compute the ratio of actual to expected reduction, $r_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}$. This value will determine if the step will be accepted or not and the update for Δ_k.

S4: Increment $k \leftarrow k + 1$ and repeat from S1.

Trust Region Basic Algorithm

Basic trust region

S0: Pick the initial point and trust region parameter x_0 and Δ_0, and set $k = 0$.

S1: Construct a quadratic model:
$m_k(s) \approx f(x_k + s)$

S2: Calculate s_k with $||s_k|| \leq \Delta_k$, so as to sufficiently reduce m_k.

S3: Compute the ratio of actual to expected reduction, $r_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}$. This value will determine if the step will be accepted or not and the update for Δ_k.

S4: Increment $k \leftarrow k + 1$ and repeat from S1.

Trust Region Basic Algorithm

Basic trust region

S0: Pick the initial point and trust region parameter x_0 and Δ_0, and set $k = 0$.

S1: Construct a quadratic model:
$m_k(s) \approx f(x_k + s)$

S2: Calculate s_k with $||s_k|| \leq \Delta_k$, so as to sufficiently reduce m_k.

S3: Compute the ratio of actual to expected reduction, $r_k = \frac{f(x_k)-f(x_k+s_k)}{m_k(0)-m_k(s_k)}$. This value will determine if the step will be accepted or not and the update for Δ_k.

S4: Increment $k \leftarrow k + 1$ and repeat from S1.

Trust Region Basic Algorithm

Basic trust region

S0: Pick the initial point and trust region parameter x_0 and Δ_0, and set $k = 0$.

S1: Construct a quadratic model:
$m_k(s) \approx f(x_k + s)$

S2: Calculate s_k with $||s_k|| \leq \Delta_k$, so as to sufficiently reduce m_k.

S3: **Compute the ratio of actual to expected reduction, $r_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}$. This value will determine if the step will be accepted or not and the update for Δ_k.**

S4: Increment $k \leftarrow k + 1$ and repeat from S1.

Trust Region Basic Algorithm

Basic trust region

S0: Pick the initial point and trust region parameter x_0 and Δ_0, and set $k = 0$.

S1: Construct a quadratic model:
$m_k(s) \approx f(x_k + s)$

S2: Calculate s_k with $||s_k|| \leq \Delta_k$, so as to sufficiently reduce m_k.

S3: Compute the ratio of actual to expected reduction, $r_k = \frac{f(x_k)-f(x_k+s_k)}{m_k(0)-m_k(s_k)}$.
This value will determine if the step will be accepted or not and the update for Δ_k.

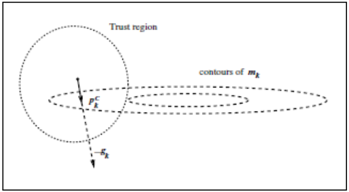
S4: Increment $k \leftarrow k + 1$ and repeat from S1.

Solving the quadratic subproblem

- Cauchy point calculation: $p_k^C = -t_k \frac{\Delta_k}{||g_k||} g_k$.
- Dogleg path: Linear combination of Cauchy point and unconstrained minimizer.
- Two dimensional subspace minimization: Search entire subspace spanned by the Cauchy point and the unconstrained minimizer
- Iterative solution to the subproblem: $p_k(\lambda) = -(B_k + \lambda I) - 1 g_k$

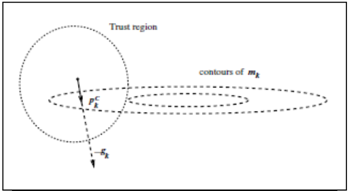
Solving the quadratic subproblem

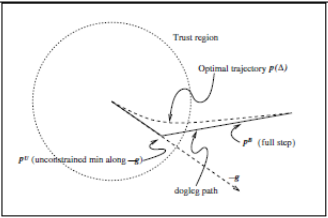
- **Cauchy point calculation:** $p_k^C = -t_k \frac{\Delta_k}{||g_k||} g_k$.
- Dogleg path: Linear combination of Cauchy point and unconstrained minimizer.
- Two dimensional subspace minimization: Search entire subspace spanned by the Cauchy point and the unconstrained minimizer
- Iterative solution to the subproblem: $p_k(\lambda) = -(B_k + \lambda I) - 1 g_k$



Solving the quadratic subproblem

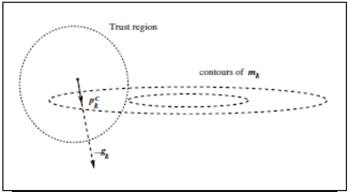
- Cauchy point calculation: $p_k^C = -t_k \frac{\Delta_k}{||g_k||} g_k$.
- **Dogleg path: Linear combination of Cauchy point and unconstrained minimizer.**
- Two dimensional subspace minimization: Search entire subspace spanned by the Cauchy point and the unconstrained minimizer
- Iterative solution to the subproblem: $p_k(\lambda) = -(B_k + \lambda I)-1 g_k$

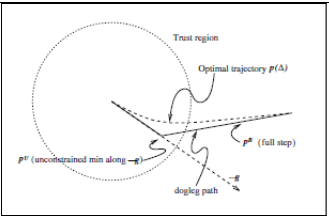




Solving the quadratic subproblem

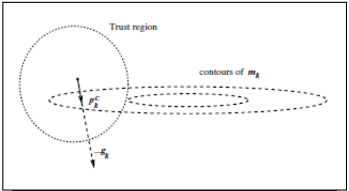
- Cauchy point calculation: $p_k^C = -t_k \frac{\Delta_k}{||g_k||} g_k$.
- Dogleg path: Linear combination of Cauchy point and unconstrained minimizer.
- **Two dimensional subspace minimization: Search entire subspace spanned by the Cauchy point and the unconstrained minimizer**
- Iterative solution to the subproblem: $p_k(\lambda) = -(B_k + \lambda I) - 1 g_k$

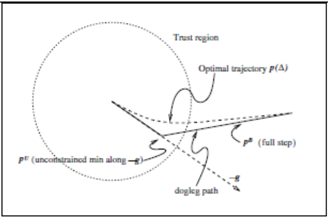




Solving the quadratic subproblem

- Cauchy point calculation: $p_k^C = -t_k \frac{\Delta_k}{||g_k||} g_k$.
- Dogleg path: Linear combination of Cauchy point and unconstrained minimizer.
- Two dimensional subspace minimization: Search entire subspace spanned by the Cauchy point and the unconstrained minimizer
- **Iterative solution to the subproblem:** $p_k(\lambda) = -(B_k + \lambda I) - 1 g_k$





Rectangular Trust Regions

Rectangular Choice

$$\min_s f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s, \quad \text{subject to: } ||s||_\infty \leq \Delta_k$$

$$\min_s f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s, \quad \text{subject to: } -\Delta_k \leq \max_i s_i \leq \Delta_k$$

- Straightforward choice for problems with bound constraints.

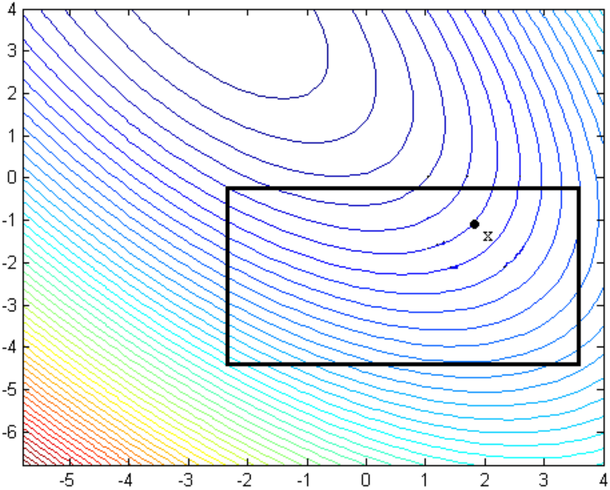
$$\min_{x \in \mathcal{R}^n} f(x) \quad \text{subject to } a_i \leq x_i \leq b_i$$

Final problem

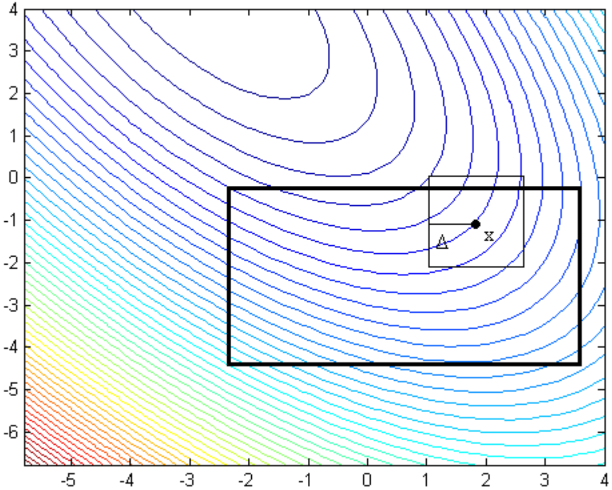
$$\begin{aligned} \min_s \quad & f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s, \\ \text{subject to:} \quad & \max(a_i - (x_k)_i, -\Delta_k) \leq s_i \leq \min(b_i - (x_k)_i, \Delta_k) \end{aligned}$$

since $x_k + s_k$ must be feasible.

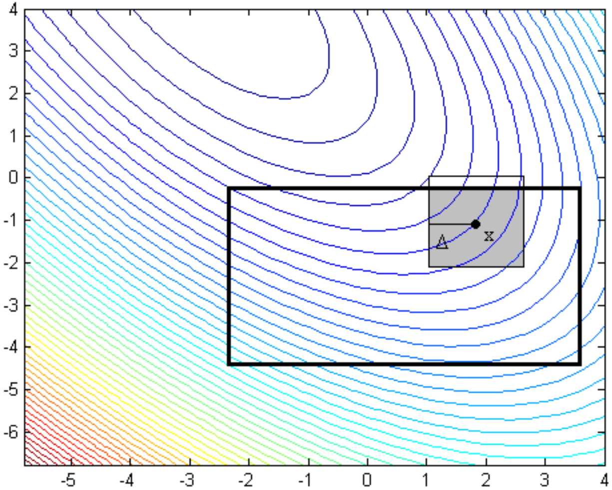
Illustration



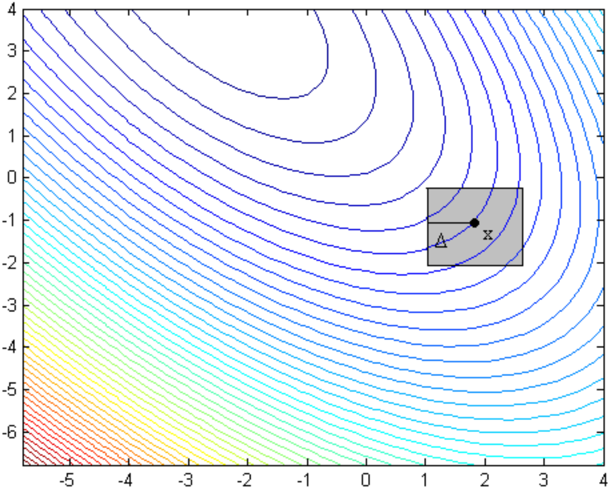
Illustration



Illustration



Illustration

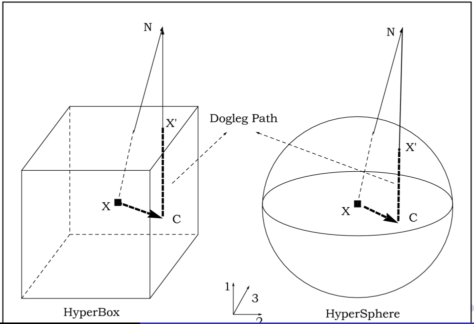


Dogleg Solution

Dogleg path

$$s(\lambda) = \begin{cases} \lambda C & \text{for } 0 \leq \lambda \leq 1 \\ C + (\lambda - 1)(N - C) & \text{for } 1 \leq \lambda \leq 2 \end{cases}$$

where $C = -\frac{g_k^T g_k}{g_k^T B_k g_k} g_k$ is the Cauchy step, and $N = -H_k^{-1} g_k$ is the Newton step, that is the unconstrained minimizer of m_k.



Dogleg Solution

- Model $m_k(s)$ decreases monotonically along the Dogleg path, assuming that H_k is positive definite.
- We can distinguish three cases:

  Case 1: $N \in T_k$
  Case 2: $C \in T_k$ and $N \notin T_k$
  Case 3: $C \notin T_k$ and $N \notin T_k$

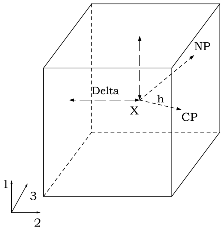
  - Original approach: Maximum feasible step along C (PC: projected Cauchy Point)
  - **Our approach: Begin from N and follow direction $B - PC$ until a bound is encountered.**

Dogleg path

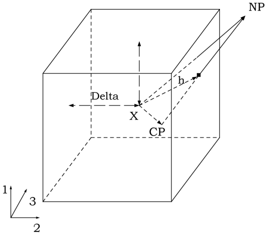
$$s(\lambda) = \begin{cases} \lambda C & \text{for } 0 \leq \lambda \leq b \\ bC + (\lambda - b)(N - bC) & \text{for } b \leq \lambda \leq 1 + b \end{cases}$$

where $b = \frac{||PC||_2}{||C||_2} \in [0, 1]$.

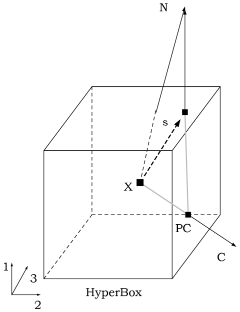
Threes Cases Illustration



Case 1



Case 2



Case

Solve:

Quadratic subproblem

$$\begin{aligned} \min_s \quad & f(x_k) + g_k^T s + \frac{1}{2} s^T B_k s, \\ \text{subject to:} \quad & \max(a_i - (x_k)_i, -\Delta_k) \leq s_i \leq \min(b_i - (x_k)_i, \Delta_k) \end{aligned}$$

using the proposed convex quadratic programming method.

Unconstrained case

Problem Name	Test Point 1						Test Point 2					
	TRUST			DOGBOX			TRUST			DOGBOX		
	It.	FC	GC	It.	FC	GC	It.	FC	GC	It.	FC	GC
ROSEN	40	47	41	37	44	38	26	31	27	27	34	28
FRE-ROT	13	40	13	14	34	14	14	40	14	14	40	14
BRO-B-S	34	43	35	34	43	35	37	50	37	37	50	38
BEA	19	20	19	18	19	18	16	19	16	18	19	20
JEN-SAM	1	7	2	1	7	2	1	17	2	1	17	2
HEL-VAL	33	43	34	30	38	30	*	*	*	*	*	*
BARD	23	42	23	20	39	20	23	41	23	22	40	22
GAUS	7	19	7	7	18	8	15	15	16	13	14	14
GULF	1	2	1	1	2	1	2	22	2	2	22	2
BOX3	37	39	38	39	40	42	52	57	53	51	57	52
POW-SIN	67	71	68	88	89	94	92	97	93	71	74	72
WOOD	36	44	36	37	46	37	24	30	25	34	43	35
KOW-OSB	33	49	33	34	49	34	41	56	41	42	62	42
BRO-DEN	37	65	37	41	69	41	42	69	42	49	83	49
OSB1	67	91	67	69	92	69	111	142	111	101	133	101
BIG-E6	44	62	44	46	69	46	41	57	41	40	58	40
OSB2	66	89	66	61	89	61	49	75	49	40	63	40
WATS	159	177	159	131	156	131	180	216	180	188	225	188
X-ROS	92	107	92	104	123	104	95	115	95	98	121	98
X-POW-S	204	218	204	221	247	231	254	274	254	204	221	204
PENI	202	226	202	172	217	172	57	81	57	38	61	38
PENII	203	241	203	270	300	271	259	300	260	253	300	254
VAR-DIM	15	21	15	25	31	25	23	28	23	24	29	24
TRIG	34	48	34	30	46	30	36	50	36	39	54	39
BR-A-LIN	19	36	19	18	34	18	1	1	1	1	1	1
DISC-INT	29	30	29	33	35	33	29	29	29	34	37	35
LIN-FR	3	5	4	2	3	2	3	4	3	2	3	2
LIN-R1	3	25	3	3	25	3	3	27	3	3	25	3

Unconstrained case

Problem Name	Test Point 1								Test Point 2							
	TRUST			BOXDOG			TOLMIN		TRUST			BOXDOG			TOLMIN	
	It.	FC	GC	It.	FC	GC	FC	GC	It.	FC	GC	It.	FC	GC	FC	GC
ROSEN	6	39	6	2	2	2	3	2	5	11	6	2	2	2	3	
FRE-ROT	39	84	39	2	2	2	3	2	1	2	1	2	2	2	3	
POW-B-S	11	29	11	2	2	2	3	2	13	32	13	3	3	3	5	
BROW-B-S	8	65	8	3	48	3	37	36	6	63	6	3	3	3	4	
BEAL	46	93	46	3	3	3	4	3	1	2	1	3	3	3	4	
JEN-SAM	1	2	1	3	3	3	5	4	1	13	2	3	3	3	6	
GAUS	15	16	15	7	18	8	14	15	56	73	56	9	9	9	31	
MEYE	63	117	63	20	47	20	25	24	-	-	-	12	12	12	23	
GULF	50	100	50	6	6	6	8	7	50	97	50	10	10	10	8	
BOX3	5	5	6	4	4	4	5	4	7	32	7	4	4	4	5	
POW-SI	-	-	-	4	4	4	5	4	-	-	-	3	3	3	4	
KOW-OSB	68	84	68	13	13	13	20	19	58	105	58	7	7	7	8	
BRO-DEN	1	9	2	3	3	3	7	6	1	12	2	3	3	3	5	
OSB1	66	115	66	250	339	250	19	18	-	-	-	11	11	11	16	
BIG-EX	53	70	53	10	11	10	19	18	30	46	30	16	32	16	27	
OSB2	73	91	73	33	53	33	59	58	58	76	58	14	30	14	22	
WATS	1	0	0	0	0	0	0	0	1	3	2	21	21	21	42	
X-ROSE	7	33	7	2	2	2	3	2	6	40	6	2	2	2	3	
X-POW-S	-	-	-	6	6	6	6	5	-	-	-	3	3	3	4	
PEN1	2	36	2	5	5	5	6	5	1	2	1	5	5	5	6	
PEN2	50	97	50	5	5	5	10	9	90	136	90	5	5	5	7	
VAR-DIM	22	82	22	10	10	10	11	10	20	70	20	10	10	10	11	
TRIG	61	78	61	19	36	19	33	32	53	99	53	11	11	11	13	
BR-A-LIN	8	41	8	3	3	3	4	3	0	0	0	0	0	0	0	
DISC-BOUN	-	-	-	20	35	20	39	38	0	0	0	0	0	0	0	
LIN-FR	46	90	46	2	2	2	3	2	45	89	45	2	2	2	3	
LIN-R1	1	5	2	11	11	11	12	11	1	7	2	11	11	11	12	
LIN-R10	1	4	2	9	9	9	10	9	1	6	2	9	9	9	10	

A Hybrid Local Search For Neural Network Training

Problem definition: Nonlinear Least-squares

$$\min_x F(x) = \frac{1}{2}\mathbf{f}^T\mathbf{f} = \frac{1}{2}\sum_{i=1}^{m} f_i^2(x),$$

$$\text{subject to : } a_i \leq x_i \leq b_i, \forall i \in I = \{1, 2, \cdots, n\}$$

Method properties:

- Employ first and second order derivatives
- Line search framework
- Use Fletcher & Xu criterion for the Sum of squares
- Application to neural network training

Large and small residuals

Derivatives

$$
\begin{aligned}
g(x) &= \nabla F(x) = J(x)\mathbf{f}(x) \\
H(x) &= \nabla^2 F(x) = J^T(x)J(x) + \sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x)
\end{aligned}
$$

Small residual case

$$
\begin{aligned}
f_i(x^*) &\simeq 0 \\
\sum_{i=1}^{m} f_i(x^*)\nabla^2 f_i(x^*) &\simeq 0 \\
H_{approx}(x^*) &= J^T(x^*)J(x^*) \text{ Gauss-Newton approximation}
\end{aligned}
$$

Large residual case

$$
f_i(x^*) \simeq 0
$$

$$
\sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) \text{ is significant}
$$

Fletcher & Xu criterion

Consider the model line search algorithm

S1. [*Initialize*] Set $k = 0, x_k =$ initial estimate

S2. [*Compute search direction*] Compute a non-zero vector p_k, by solving $\tilde{H}_k p_k = -g_k$

S3. [*Compute step length*] $\lambda_k = argmin_\lambda f(x_k + \lambda p_k)$

S4. [*Update the estimate of the minimum*] $x_{k+1} = x_k + \lambda_k + p_k$

The matrix $\tilde{H}_k$:

- Positive definite correction of hessian matrix $H(x_k) \rightarrow$ Newton's method
- Positive definite approximation $J^T(x_k)J(x_k) \rightarrow$ Gauss-Newton method.

- For large residuals use Newton's method
- For small residuals use Gauss-Newton approximation

Fletcher & Xu criterion

- Instead of choosing an algorithm (Newton or Gauss-Newton)
- Unify both approaches using a switching criterion
- The switching criterion is based on relative decrease of $F(x)$
- Originally proposed for quasi-Newton instead of Newton directions

$$\lim_{k\to\infty} \frac{F_k - F_{k+1}}{F_k} = \begin{cases} 0 & \text{for the LRP,} \\ 1 & \text{for the ZRP.} \end{cases}$$

Modification of the algorithm

S2. Compute search direction.

Set $B_k = \begin{cases} \nabla^2 f_k & \text{if } f_{k-1} - f_k / f_{k-1} < \epsilon, \\ J_k^T J_k & \text{otherwise.} \end{cases}$

Solve $B_k p_k = -g_k$ to get the search direction p_k

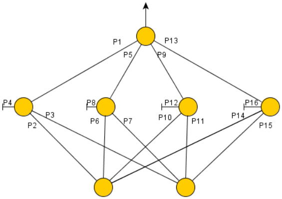
Neural network training

Let $N(x, p)$ denote an ANN with input vector x and weights p. In our case this will be a perceptron with one hidden layer with sigmoidal units and linear output activation, i.e.

$$N(x,p) = \sum_{i=1}^{h} p_{i(n+2)-(n+1)} \sigma \left(\sum_{k=1}^{n} p_{i(n+2)-(n+1)+k} x_k + p_{i(n+2)} \right)$$

where:

- $x_i, \ \forall i = 1, \cdots, n$ are the components of the input vector $x \in R^{(n)}$.
- $p_i, \ \forall i = 1, \cdots, h(n+2)$ are the components of the weight vector p.
- h, denotes the number of hidden units.
- $\sigma(z) \equiv (1 + exp(-z))^{-1}$ is the sigmoid used as activation.



The training of the ANN to existing data is performed by minimizing the following "Error function":

$$f(p) = \frac{1}{2} \sum_{K=1}^{M} r_K^2 \equiv \frac{1}{2} \sum_{K=1}^{M} [N(x_K, p) - y_K]^2$$

Neural network training

- Nonlinear Least-square case $f_i(x) = N(x_i, p) - y_i$
- Many minima having zero and nonzero Error function value
- Large residual and small residual cases
- Analytical first and second order derivatives

$$\frac{\partial^2 N(x_K, p)}{\partial p_{l(n+2)-(n+1)+m} \partial p_{r(n+2)-(n+1)+s}} =$$

$l = r$	$m = 0$	$s = 0$	0
		$s = 1 \dots n$	$\sigma'(Y_j)x_s$
		$s = n+1$	$\sigma'(Y_j)$
	$m = 1 \dots n$	$s = 0$	$\sigma'(Y_j)x_m$
		$s = 1 \dots n$	$p_{l(n+2)-(n+1)} x_m x_s \sigma''(Y_j)$
		$s = n+1$	$p_{l(n+2)-(n+1)} x_m \sigma''(Y_j)$
	$m = n+1$	$s = 0$	$\sigma'(Y_j)$
		$s = 1 \dots n$	$p_{l(n+2)-(n+1)} x_s \sigma''(Y_j)$
		$s = n+1$	$p_{l(n+2)-(n+1)} \sigma''(Y_j)$
$l \neq r$	$m = 0 \dots n+1$	$s = 0 \dots n+1$	0

$$Y_j = \sum_{k=1}^{n} p_{j(n+2)-(n+1)+k} x_k + p_{j(n+2)}$$

General test functions

Test name	BFGS	LEVE	Hybrid Newton
	Func Eval/Iter	Func Eval/Iter	Func Eval/Iter(Gauss Steps)
ROSENBROCK	1.986 *10-16 80/12	0.000 10//3	3.958*10-16 16/2(1)
FREUDENSTEIN AND ROTH	7.183 *10-16 77/14	7.888 *10-31 28//9	48.984 64/10(2)
POWELL BADLY SCALED	Acc Stop	1.232 * 1032 202/59	1.102 * 10-8 23/4(2)
BROWN BADLY SCALED	7,17244E+11 40/2	2.549 * 10-29 50/17	1.139*10-14 1198/125(63)
BEALE	1.359*10-18 161/24	0.452 5545/1700	0.452 10016/1269(1)
JENNRICH AND SAMPSON	2020 38/1	259.58 74/24	2020 57/10(8)
HELICAL VALEY	3.024*10-34 183/24	1.271*10?57 30-8	5.933*10-38 110/13(10)
BARD	8.214*10-3 142/18	8.214*10-3 46/9	8.214*10-3 148/17(6)
GAUSSIAN	1.128*10-8 256/32	0.564 21/5	1.128*10-8 240/27(9)
MEYER	Acc Stop	87.94 28813/6708	8477691 10006/1005(6)
GULF	3.849*10?2 0 Iterations The gradient criterion is satisfied		
BOX 3-D	1.036*10-24 102/12	2.773*10-32 92/20	5.718*10-22 118/5(4)
POWELL SINGULAR	7.263*10?24 712/75	1.609*10-63 367/70	7.222*10-32 1471/73(62)
WOOD	1.187*10-17 587/62	0.000 36/7	1.187*10-17 169/18(13)

General test functions

Test name	BFGS Func Eval/Iter	LEVE Func Eval/Iter	Hybrid Newton Func Eval/Iter(Gauss Steps)
KOWALIK AND OSBORNE	3.075*10-4 665/68	1.027*10?3 569/110	1.027*10?3 10001/833(5)
BROWN AND DENNIS	85822.22 254/21	85822.22 358/69	85822.22 207/18(5)
OSBORNE 1	1.106 50/4	1.106 51/7	1.106 60/5(1)
BIGGS EXP6	0.306 228/17	0.180 16730/2307	Acc Stop
OSBORNE 2	1.790 549/17	1.790 171/14	1.790 466/7(1)
WATSON	2.829*10-13 7963/184	2.836*10-3 8210/39	868908 424/6(2)
EXTENDED ROSENBROCK	1.998*10-15 8976/406	0.000 46/5	1.987*10-15 231/10(9)
EXTENDED POWELL SINGULAR	3.705*10-16 3739/147	3.112*10-68 952/72	7.928*10-32 4166/87(70)
PENALTY I	2.249*10-5 1886/195	2.249*10-5 179/32	2.249*10-5 918/90(10)
PENALTY II	9.376*10-6 12825/1351	9.376*10-6 160/29	Acc Stop
VARIABLY DIMENSIONED	2.674*10-30 715/33	0.000 155/14	0.000 85/3(2)
TRIGONOMETRIC	4.224*10-5 1749/81	8.788*10-4 277/23	2.795*10-5 1120/48(17)
BROWN ALMOST LINEAR	1.316*1013 50/0/1	1.000 179/16	9.478*10-30 682/25(21)
DISCRETE BOUNDARY PR	9.358*10-21 751/35	2.503*10-33 90/9	8.962*10-21 203/9(7)
DISCRETE INTEGRAL EQ	1.116*10-22 574/27	3.229*10-33 90/9	1.034*10-22 222/10(8)

Neural network training: Large residual case

$n = 2$ hidden nodes $h = 5$ and training data $M = 100$.

Table: LRP: Minimum No 1

Method	Iterations	Function/Gradient calls
Hybrid Newton	126	357
Hybrid BFGS	335	652
Newton	719	1000
Gauss-Newton	1000	3000
Tolmin	174	252
Conjugate Gradient	1480	6000
Minimum value		18.486

Table: LRP: Minimum No 2

Method	Iterations	Function/Gradient calls
Hybrid Newton	20	64
Hybrid BFGS	33	100
Newton	35	57
Gauss-Newton	150	301
Tolmin	74	107
Conjugate Gradient	77	302
Minimum value		19.266

Neural network training: Small residual case

$n = 2$ hidden nodes $h = 5$ and training data $M = 100$.

Table: SRP: Minimum No 1

Method	Iterations	Function/Gradient calls	Minimum reached
Hybrid Newton	115	275	0
Hybrid BFGS	363	487	0
Newton	316	364	0
Gauss-Newton	257	296	0
Tolmin	513	697	0
Conjugate Gradient	2318	10000	1.0768
Minimum value			0

Table: SRP: Minimum No 2

Method	Iterations	Function/Gradient calls	Minimum reached
Hybrid Newton	599	1000	0.0961
Hybrid BFGS	623	713	0.0101
Newton	759	1000	2.3282
Gauss-Newton	734	1000	0.4374
Tolmin	710	1000	0.0733
Conjugate Gradient	965	4000	1.2228
Minimum value			0