A New Error Function and Its Application in Distance Geometry Problem

Zhenli Sheng (盛镇醴)

email: szl@lsec.cc.ac.cn

Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences

joint work with Prof. Ya-xiang Yuan

January 8, 2013

seminar talk

Outline

Outline

1. Problem introduction

Distance Geometry Problem

Find the coordinate vectors $x_1, x_2, \ldots, x_n$ that satisfy several given distances between them. Mathematically, this problem can be stated as follow,

Find $x_1, x_2, \ldots, x_n$, such that

$$\|x_i - x_j\| = d_{i,j}, \quad (i,j) \in S.$$

or

$$l_{i,j} \leq \|x_i - x_j\| \leq u_{i,j}, \quad (i,j) \in S.$$

- The data given may have some errors.
- This problem can be formulated as global optimization problem.
- It has many applications.

Application I: Graph Realization

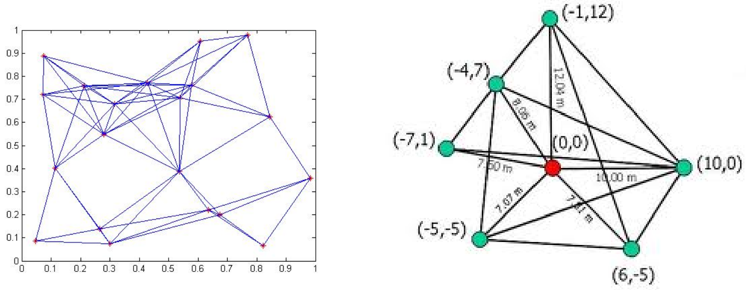


Figure 1 : Graph Realization in 2D

Given a graph G=(V,E), each edge has a weight.

Application II: Protein Structure Determination

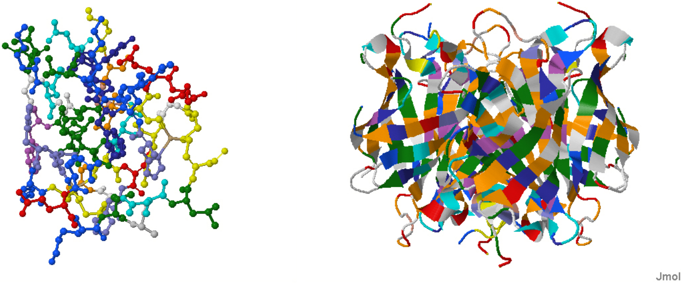


Figure 2 : Two proteins: 1PTQ and 1HQQ, in different display ways

Measure distances: NMR, X ray crystallography.

Application III: Sensor Network Localization

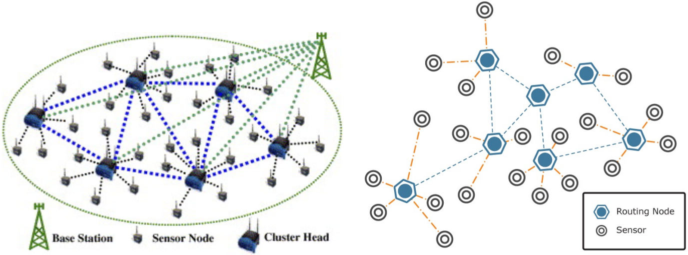


Figure 3 : Illustration of wireless sensor networks

Outline

Related works

- Matrix Decomposition Method (Blumenthal-1953, Torgerson-1958)
- The Embedding Algorithm (Crippen-Havel-1988)
- Global Smoothing Algorithm (Moré-Wu-1997)
- Geometric Buildup Method (Dong-Wu-2002, Sit-Wu-Yuan-2009)
- SDP Relaxation Method (Biswas-Toh-Ye-2007)
- ...

Matrix Decomposition Method

DG problem with full set of exact distances

Given a full set of distances, $d_{i,j} = \|x_i - x_j\|, \quad i,j = 1,2,\ldots,n.$

- Set $x_n = (0,0,0)^{\mathrm{T}}$, we have

$$\begin{aligned} d_{i,j}^2 &= \|x_i - x_j\|^2 \\ &= \|x_i\|^2 - 2x_i^{\mathrm{T}} x_j + \|x_j\|^2 \\ &= d_{i,n}^2 - 2x_i^{\mathrm{T}} x_j + d_{j,n}^2, \qquad i,j = 1,2,\ldots,n-1 \end{aligned} \tag{1}$$

- Define $X = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ and $D = \{(d_{i,n}^2 - d_{i,j}^2 + d_{j,n}^2)/2 : i,j = 1,2,\ldots,n-1\}$, (1) $\Rightarrow XX^{\mathrm{T}} = D$.
- Let $D = U\Sigma U^{\mathrm{T}}$, $V = U(:,1:3)$ and $\Lambda = \Sigma(1:3,1:3)$. Then $X = V\Lambda^{1/2}$ solves the problem. [Eckart and Young 1936]

Unconstrained optimization: Error functions

- stress function

$$Stress(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in S} (\|x_i - x_j\| - d_{i,j})^2,$$

- smoothed stress function

$$SStress(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in S} (\|x_i - x_j\|^2 - d_{i,j}^2)^2,$$

- generalized stress function

$$GStress(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in S} \min{}^2\{\frac{\|x_i - x_j\|^2 - l_{i,j}^2}{l_{i,j}^2}, 0\} + \max{}^2\{\frac{\|x_i - x_j\|^2 - u_{i,j}^2}{u_{i,j}^2}, 0\}.$$

Goal and Difficulties

- Goal: minimize the chosen error function to the global minimizer—zero
- Difficulties: NP-hard in general
  - too many local minimizers
  - possibly nonsmooth
  - large-scale problems

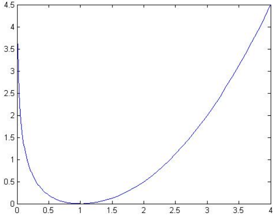
Outline

3. Our proposed error function and algorithm

Our proposed error function

Define h: $\mathbb{R}_{++} \to \mathbb{R}$ as below,

$$h(x) = \begin{cases} \frac{1}{2}(x-1)^2, & x \geq 1, \\ x - (1 + ln(x)), & x < 1. \end{cases}$$



Theorem 3.1

h(x) is twice continuously differentiable in $(0, +\infty)$*, and it achieves its minimum 0 at 1.*

But... why this function?

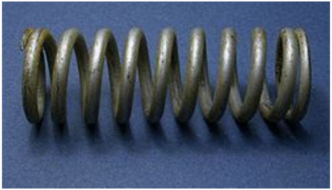


Figure 4 : Hooke's law models the properties of springs for small changes in length

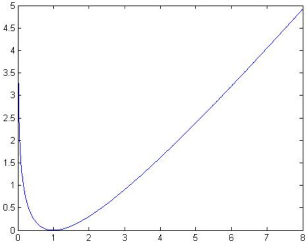
Force function:

$$F = \begin{cases} x - 1, & x \geq 1, \\ 1 - \frac{1}{x}, & x < 1. \end{cases}$$

then the energy function is h(x).

Another proposal:

$h(x) = x - (1 + ln(x))$



Modeling and Solution idea

Using our error function, the distance geometry problem can be formulated as

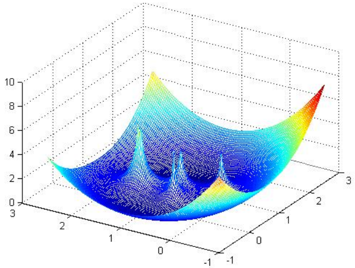
$$\min \quad f(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in S} h\left(\frac{\|x_i - x_j\|}{d_{i,j}}\right). \qquad (2)$$

- Observation:
  - huge items in the objective function
  - variables are mixed together $\Rightarrow$ not easy to calculate Hessian matrix
- Solution idea:
  - use first-order algorithm — "alternative direction method" to solve it $\Rightarrow$ fixed the others, adjust $x_i (i = 1, \ldots, n)$ in turn.

Trust region subproblem

Let $\overline{f}(x) = \sum_{j \in N(i)} h(\frac{\|x - x_{\boldsymbol{j}}\|}{d_{\boldsymbol{i},\boldsymbol{j}}})$, where $N(i)$ is the neighbourhood of point i, then the "trust region subproblem" at each iteration is as following,

$$\begin{array}{lll} \min_{s} & s^{\mathrm{T}} \nabla \overline{f}(x_i) + \frac{1}{2} s^{\mathrm{T}} \nabla^2 \overline{f}(x_i) s & \triangleq q(x) \\ \text{s.t.} & \|s\| \leq \Delta. & \end{array} \tag{3}$$



Trust region subproblem (Cont'd)

Theorem 3.2

Define $\overline{h} : \mathbb{R}^d \to \mathbb{R}, \overline{h}(x) = h(\frac{\|x-a\|}{d})$, *where* $a \in \mathbb{R}^d$ *and* $d \in \mathbb{R}$ *are constants, then*

- $$\nabla \overline{h}(y) = \frac{y}{d\|y\|} - \frac{y}{\|y\|^2},$$

  and

  $$\begin{aligned} \nabla^2 \overline{h}(y) &= -\frac{yy^{\mathrm{T}}}{d\|y\|^3} + \frac{2yy^{\mathrm{T}}}{\|y\|^4} + (\frac{1}{d\|y\|} - \frac{1}{\|y\|^2})I \\ &= (\frac{1}{d\|y\|} - \frac{1}{\|y\|^2})(I - \frac{yy^{\mathrm{T}}}{\|y\|^2}) + \frac{yy^{\mathrm{T}}}{\|y\|^4}, \end{aligned}$$

  where

  $$y = x - a \text{ and } I \text{ is the identity matrix.}$$

- *if* $\|y\| \geq d, \nabla^2 \overline{h}(y)$ *is positive definite, otherwise it can be negative definite.*

Stopping criteria

1. the objective function or the improvement is small enough, i.e.

$$f_k < tol \quad or \quad |f_k - f_{k-1}| < tol$$

2. the norm of the gradient is small enough, i.e.

$$\|\nabla f(x)\| < MinNorm$$

   where tol, $MinNorm$ are some given small numbers, for instance, 10^{-5}.

3. the outer iteration achieves its maximum permitted number, i.e. $k < MaxItr$

♠ One of the above three criteria is satisfied, the algorithm will stop.

Algorithm framework I

Algorithm 1: Trust Region Error Minimization Method

Initialization: Give initial points and set some parameters

while *stopping criteria not satisfied* **do**

  for *i=1:n* **do**

    Solve trust region subproblem (3) to obtain s_i, let

$$r_i = \frac{\overline{f}(x_i) - \overline{f}(x_i + s_i)}{q(x_i) - q(x_i + s_i)}$$

    According to r_i to determine to accept s_i or not, and adjust Δ_i;

  end

end

Nonmonotone Newton step

- Let $\overline{f}(x)$ be defined as before, then "Newton step" can be given as below,

$$d_i^N = -(\nabla^2 \overline{f}(x_i))^{-1} \nabla \overline{f}(x_i)$$

  Let search direction be

$$d_i = \begin{cases} d_i^N & \textit{if } \nabla^2 \overline{f}(x_i) \textit{ is positive definite}, \\ -\nabla \overline{f}(x_i) & \textit{otherwise}, \end{cases} \tag{4}$$

  which is a descent direction.

- Search for stepsize α_i by backtracking (start at 1), such that

$$f(x_i + \alpha_i d_i, x_{-i}) < MaxF + \frac{1}{2} \alpha_i d_i^{\mathrm{T}} \nabla \overline{f}(x_i) \tag{5}$$

  where *MaxF* is the maximum objective function value of lastest M step.

Algorithm framework II

Algorithm 2: Nonmonotone Newton Error Minimization Method

Initialization: Give initial points and set some parameters

while *stopping criteria not satisfied* **do**

  for *i=1:n* **do**

    Calculate search direction d_i by (4);

    Calculate stepsize α_i by (5);

    Set $x_i \leftarrow x_i + \alpha_i d_i$

  end

end

Algorithm framework III

Algorithm 3: Trust Region-Newton Error Minimization Method

1. Apply **Algorithm 1** for K iterations,
2. Apply **Algorithm 2** then.

K can be set adaptively.

Outline

Experiments construction

- Uniformly sample nodes in square [0,1]x[0,1].
- Generate distance matrix by *disk graph model*, usually set cutoff=0.2, thus about 10% distance are available.
- Generate initial points with 20% perturbation, more specifically,

$$X0 = X. * (1 + 0.2 * (rand(n, 2) - 0.5)).$$

- Use function value and cost time as the compare criteria.

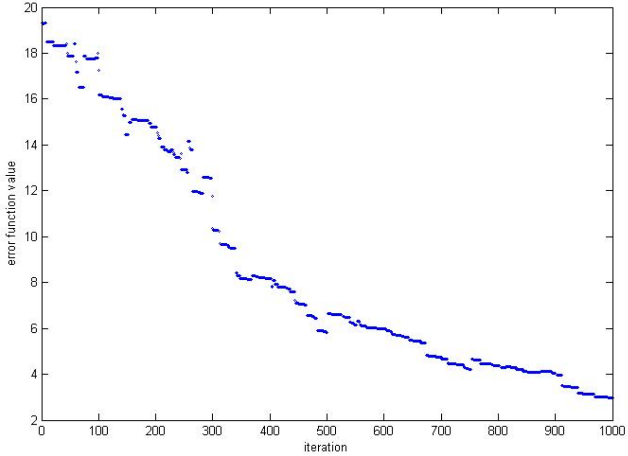
Monotone VS Nonmonotone stepsize

$cotoff = 0.2, exact\ \ distance, perturbation = 20\%, MaxItr = 100$							
$n = 100, tol = 10^{-3}$				$n = 200, tol = 10^{-3}$			
M	iter	fval	t(s)	M	iter	fval	t(s)
1	17	0.080	3.37	1	100	5.206	40.40
5	35	0.012	3.98	5	100	4.170	34.17
20	32	0.014	3.86	20	100	1.893	34.36
50	35	0.012	4.00	50	100	0.263	32.44
100	30	0.013	3.76	100	100	0.112	31.63
1	36	0.474	5.30	1	84	0.430	31.03
5	31	0.462	4.46	5	65	0.417	24.25
20	30	0.459	4.42	20	63	0.417	23.88
50	28	0.456	4.33	50	63	0.416	23.82
100	28	0.456	4.33	100	63	0.415	23.81

Monotone VS Nonmonotone stepsize

$cotoff = 0.2$, $exact\ distance$, $perturbation = 20\%$, $MaxItr = 100$							
$n = 500$, $tol = 10^{-2}$				$n = 1000$, $tol = 10^{-1}$			
M	iter	fval	t(s)	M	iter	fval	t(s)
1	100	27.840	193.64	1	100	476.719	949.58
5	100	18.176	158.75	5	57	495.770	550.68
20	100	27.614	162.99	20	65	288.151	574.78
50	100	22.696	158.43	50	90	112.026	667.39
100	100	29.526	157.47	100	75	127.622	5146.84
1	100	56.317	209.39	1	57	395.741	722.18
5	100	53.942	177.58	5	48	316.193	606.12
20	100	25.786	172.77	20	41	203.620	524.87
50	100	26.246	168.34	50	41	57.026	495.40
100	100	23.701	169.85	100	41	97.319	520.48

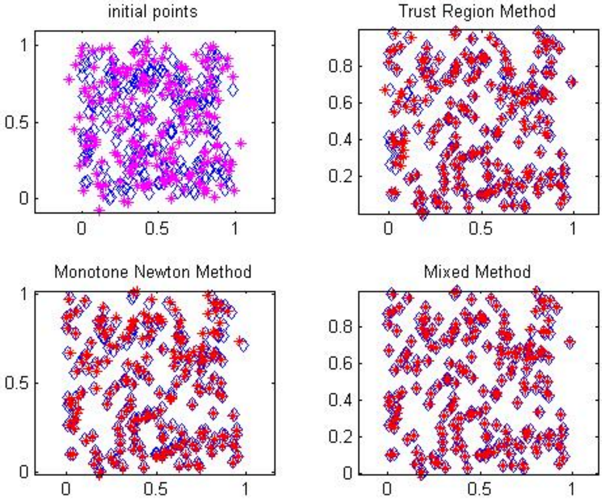
Nonmonotone Newton



Trust region VS Newton

$cotoff = 0.2, exact\ distance, perturbation = 20\%, MaxItr = 150$							
$n = 200, tol = 10^{-3}$				$n = 500, tol = 10^{-2}$			
Alg	iter	fval	t(s)	Algo	iter	fval	t(s)
Alg1	150	12.234	40.35	Alg1	150	167.611	230.60
Alg2	150	1.234	38.17	Alg2	150	221.197	312.59
Alg3	129	0.531	26.97	Alg3	150	11.380	197.97
Alg1	150	11.500	39.91	Alg1	150	302.333	222.11
Alg2	150	5.305	37.93	Alg2	150	119.509	273.53
Alg3	150	0.269	30.68	Alg3	150	24.185	186.34
Alg1	150	27.146	37.68	Alg1	150	273.636	226.34
Alg2	150	9.764	36.75	Alg2	93	188.148	185.29
Alg3	150	0.365	30.10	Alg3	150	49.415	206.23

Trust region VS Newton



Trust region VS Newton

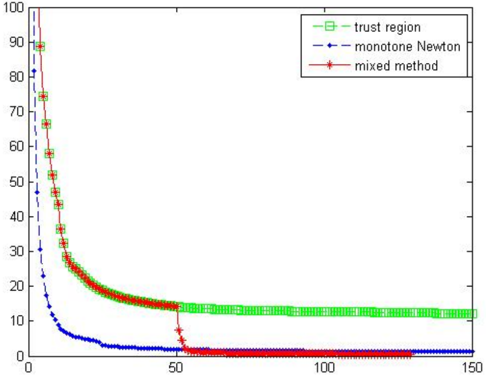
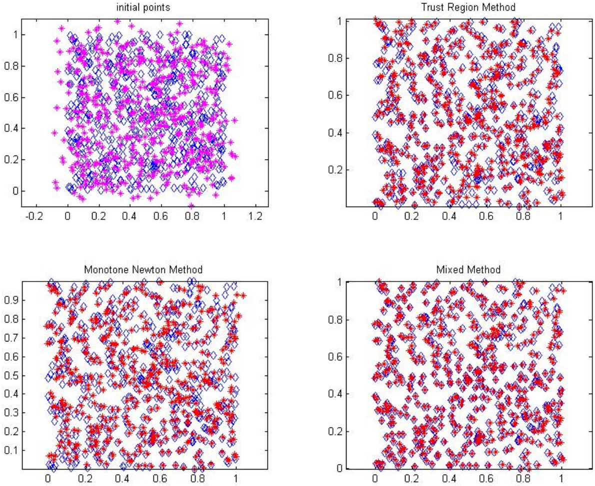


Figure 5 : An typical example: 200 nodes, exact distance, 20% perturbation

Trust region VS Newton



Trust region VS Newton

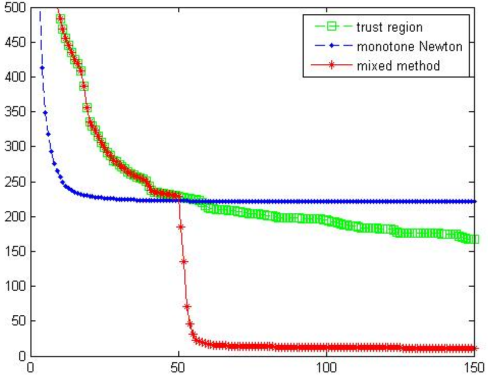


Figure 6 : An typical example: 500 nodes, exact distance, 20% perturbation

Hybrid adaptively

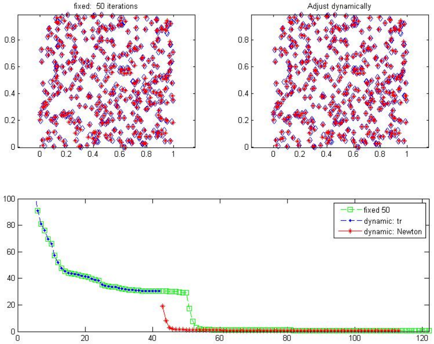


Figure 7 : An typical example: 200 nodes, exact distance, 20% perturbation

Hybrid adaptively

method	k'	iter	fval	time
fixed	50	122	0.789	25.06
dynamic	42	113	0.671	23.49

set up: 200 nodes, cutoff=0.2, exact distances, 20% perturbation, $tol = 10^{-3}$. switch criterion: k=min{50, k'}, where k' is the smallest number sunch that

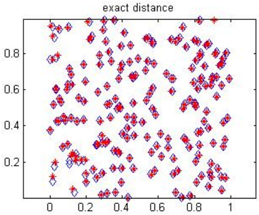
$$fval(i-1) - fval(i) < 100 * tol.$$

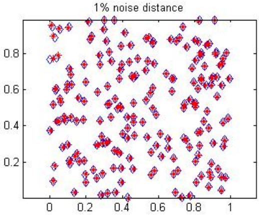
- Performance differ slightly with different parameters, usually is not worse that fixed method.

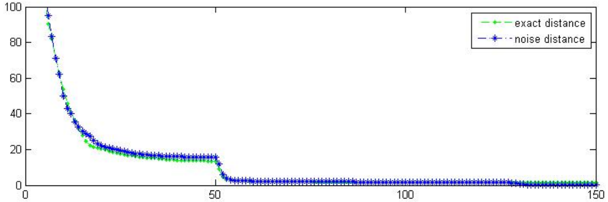
exact VS noise distances

$cotoff = 0.2, perturbation = 20\%, MaxItr = 150, tol = 10^{-3}$							
$n = 200, noise = 1\%$				$n = 200, noise = 5\%$			
dist	iter	fval	t(s)	dist	iter	fval	t(s)
exact	150	1.548	30.52	exact	150	0.111	31.22
noise	150	0.304	30.64	noise	150	1.665	30.93
exact	137	0.137	29.65	exact	148	1.060	29.94
noise	136	0.677	30.55	noise	150	1.167	30.32
$n = 200, noise = 10\%$				$n = 200, noise = 20\%$			
exact	143	0.037	30.05	exact	143	0.420	28.71
noise	150	0.734	31.07	noise	150	5.265	30.06
exact	150	0.516	29.57	exact	109	0.057	22.83
noise	150	2.134	29.68	noise	77	5.068	18.23

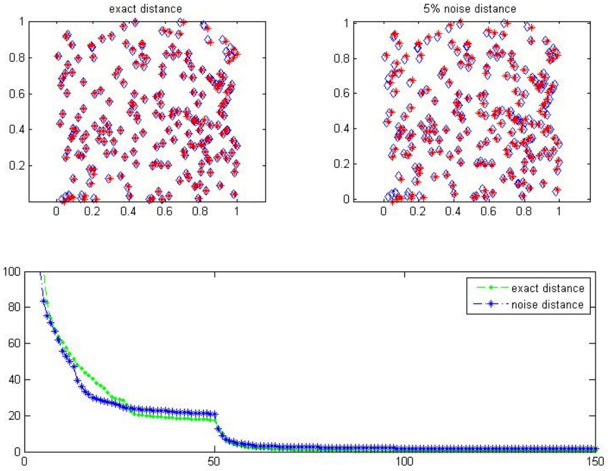
Trust region VS Newton



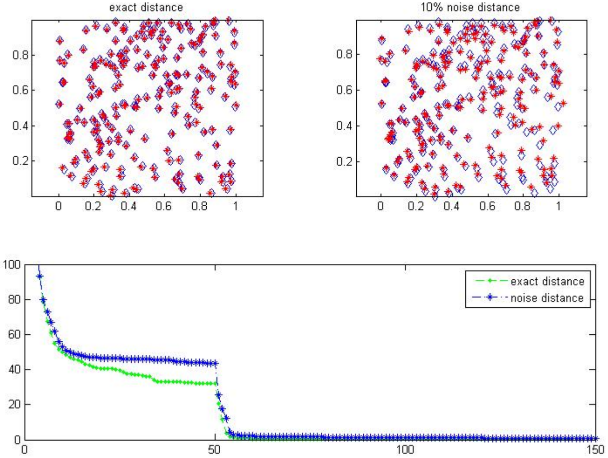




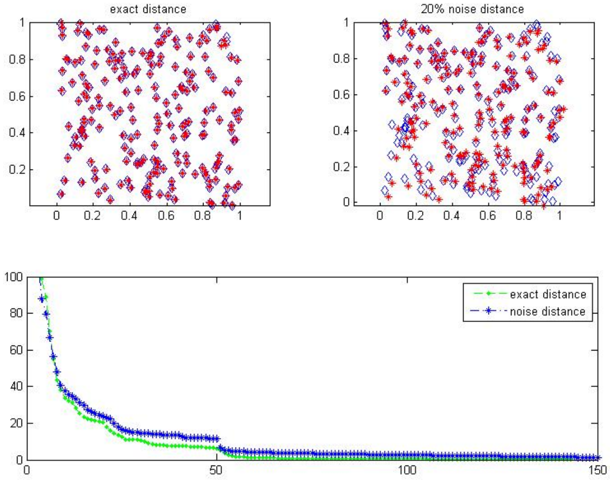
Trust region VS Newton



Trust region VS Newton



Trust region VS Newton



Outline

5. Conclusions and ongoing works

A generalized DG problem

DG problem with distance bounds

Given the lower bounds $l_{i,j}$ and upper bounds $u_{i,j}$, the problem can be formulated as (Sit-Wu-2011):

$$
\begin{aligned}
\max_{x_i, r_i} \quad & \sum_{i=1}^{n} r_i \\
\text{s.t.} \quad & \|x_i - x_j\| + r_i + r_j \leq u_{i,j} \\
& \|x_i - x_j\| - r_i - r_j \geq l_{i,j} \qquad \forall (i,j) \in S \\
& r_i \geq 0, \qquad i = 1, 2, \ldots, n.
\end{aligned}
$$

??? Gradient calculation

I try to write the problem in a compact form (in matrix/vector).

Define the coefficient matrix $A \in \mathbb{R}^{|S| \times n}$, for example (a clique with four points)

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

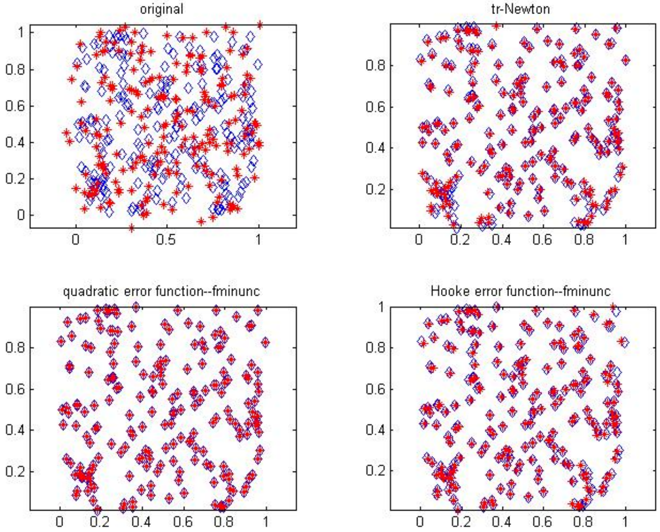
and $X = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$, $d = (d_{ij})$ which is a column vector, then

$$AX = \begin{pmatrix} \cdots \\ x_i - x_j \\ \cdots \end{pmatrix}.$$

Then ??? ...... The problem is that x_i is a vector, rather than a scalar.

how to compare functions?

- is it possible to "count" how many local minimizers a function have (even though roughly)?
- numerical ways?
- figure intuition?


original
tr-Newton
quadratic error function--fminunc
Hooke error function--fminunc
0
0.2
0.4
0.5
0.6
0.8
1

Conclusions and Future work

- What we have done:
  - proposed a novel error function
  - based on the proposed function, designed an efficient algorithm to solve the distance geometry problem
  - finished some preliminary numerical experiments, which seems promising, especially in the noise case
- Future work:
  - theoretical convergence analysis of the algorithm
  - generalize the error function to handle the "bound" case
  - compare the algorithm with the existing ones

Q & A

Thank you for your attention!