Geometria molecolare con distanze Euclidee

Un importante problema legato alla conformazione molecolare è quello del Molecular Distance Geometry Problem (MDGP). Considerando una molecola di N atomi, è dato un insieme S di coppie $\{i, j\}$ di atomi e un insieme D delle relative distanze:

$$D = \{d_{ij} \mid \{i, j\} \in S\}.$$

Si vuole trovare la posizione in $\mathbb{R}^3$ degli N atomi in modo che le distanze in D siano rispettate. Si scriva un programma Matlab che risolva il problema, e si trovi la soluzione dell'istanza seguente (con $N = 4$):

(i, j)	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
d_{ij}	1.526	2.491389536	3.837572036	1.526	2.491389535	1.526

Figura 1: Configurazione ottimale per l'istanza del problema.

Un riferimento bibliografico per questo esercizio è:

> J.M. Yoon, Y. Gad, Z. Wu, *Mathematical modeling of protein structure using distance geometry*, Technical report TR00-24, DCAM, Rice University, Houston, 2000.

che può essere scaricato da: `http://www.caam.rice.edu/caam/trs/tr00.html#TR00-24`.

Soluzione

Indichiamo con $\mathbf{x}^i = (x_1^i, x_2^i, x_3^i)$ la posizione dell'i-esimo atomo per $i \leq N$. Sappiamo che devono valere i seguenti vincoli:

$$\forall \{i,j\} \in D \ (||\mathbf{x}^i - \mathbf{x}^j|| = d_{ij}), \tag{1}$$

dunque possiamo ottenere la soluzione cercata minimizzando la somma dei quadrati degli scarti $||\mathbf{x}^i - \mathbf{x}^j|| - d_{ij}$. Si consideri, tuttavia, che l'espressione $||\mathbf{x}^i - \mathbf{x}^j||$ contiene una norma Euclidea, che coinvolge una radice quadrata — per evitare questo termine nonlineare conviene minimizzare la somma dei quadrati degli scarti quadratici $||\mathbf{x}^i - \mathbf{x}^j||^2 - d_{ij}^2$. Il problema può essere modellato con $\min f(x)$, dove

$$f(x) = \sum_{\{i,j\} \in S} (||\mathbf{x}^i - \mathbf{x}^j||^2 - d_{ij}^2)^2.$$

Si noti che l'MDGP è in realtà un problema di soddisfazione di vincoli senza alcuna funzione obiettivo; abbiamo però riformulato il problema in modo esatto a un problema nonlineare senza vincoli.

Espandendo uno qualsiasi dei termini della sommatoria $(||\mathbf{x}^i - \mathbf{x}^j||^2 - d_{ij}^2)^2$ si ottiene il termine

$$\begin{aligned}
t = \ & - \ 4\,x_2^i\,x_2^j\,{x_3^i}^2 - 4\,x_1^i\,x_1^j\,{x_2^i}^2 - 4\,x_1^i\,x_1^j\,{x_2^j}^2 - 4\,x_1^i\,x_1^j\,{x_3^i}^2 + 4\,x_1^i\,x_1^j\,d_{ij} \\
& - \ 4\,{x_1^i}^2\,x_2^i\,x_2^j - 4\,{x_1^j}^2\,x_2^i\,x_2^j - 4\,{x_2^i}^2\,x_3^i\,x_3^j - 4\,x_1^i\,x_1^j\,{x_3^j}^2 - 4\,{x_1^j}^2\,x_3^i\,x_3^j \\
& + \ 4\,x_2^i\,x_2^j\,d_{ij} - 4\,x_2^i\,x_2^j\,{x_3^j}^2 + 4\,x_3^i\,x_3^j\,d_{ij} - 4\,{x_2^j}^2\,x_3^i\,x_3^j - 4\,{x_1^i}^2\,x_3^i\,x_3^j + d_{ij}^2 \\
& + \ 8\,x_1^i\,x_1^j\,x_2^i\,x_2^j + 8\,x_1^i\,x_1^j\,x_3^i\,x_3^j + 8\,x_2^i\,x_2^j\,x_3^i\,x_3^j + {x_1^i}^4 + {x_1^j}^4 + {x_2^i}^4 \\
& + \ {x_2^j}^4 + {x_3^i}^4 + {x_3^j}^4 - 4\,{x_1^i}^3\,x_1^j + 6\,{x_1^i}^2\,{x_1^j}^2 + 2\,{x_1^i}^2\,{x_2^i}^2 + 2\,{x_1^i}^2\,{x_2^j}^2 \\
& + \ 2\,{x_1^i}^2\,{x_3^i}^2 + 2\,{x_1^i}^2\,{x_3^j}^2 - 2\,{x_1^i}^2\,d_{ij} - 4\,x_1^i\,{x_1^j}^3 + 2\,{x_1^j}^2\,{x_2^i}^2 + 2\,{x_1^j}^2\,{x_2^j}^2 \\
& + \ 2\,{x_1^j}^2\,{x_3^i}^2 + 2\,{x_1^j}^2\,{x_3^j}^2 - 2\,{x_1^j}^2\,d_{ij} - 4\,{x_2^i}^3\,x_2^j + 6\,{x_2^i}^2\,{x_2^j}^2 + 2\,{x_2^i}^2\,{x_3^i}^2 \\
& + \ 2\,{x_2^i}^2\,{x_3^j}^2 - 2\,{x_2^i}^2\,d_{ij} - 4\,x_2^i\,{x_2^j}^3 + 2\,{x_2^j}^2\,{x_3^i}^2 + 2\,{x_2^j}^2\,{x_3^j}^2 - 2\,{x_2^j}^2\,d_{ij} \\
& - \ 4\,{x_3^i}^3\,x_3^j + 6\,{x_3^i}^2\,{x_3^j}^2 - 2\,{x_3^i}^2\,d_{ij} - 4\,x_3^i\,{x_3^j}^3 - 2\,{x_3^j}^2\,d_{ij},
\end{aligned}$$

che per semplice ispezione non può essere dimostrato convesso. Dobbiamo perciò assumere che l'intera funzione obiettivo sia non convessa, e quindi è necessario utilizzare un approccio di ottimizzazione globale.

Tra gli algoritmi di ottimizzazione globale, uno molto conosciuto e molto semplice da implementare è il Multistart; si tratta di un algoritmo euristico stocastico, nel senso che non garantisce l'ottimalità globale della soluzione ma esibisce proprietà di convergenza stocastica (nel senso che converge all'ottimo globale con probabilità 1 in tempo infinito). Questo caso, tuttavia, è particolare. Nella soluzione ottima del problema tutti i vincoli (1) sono soddisfatti, e quindi il valore della funzione obiettivo è zero; possiamo quindi valutare a ogni iterazione dell'algoritmo di Multistart quanto siamo lontani dall'ottimo, e

decidere di farlo terminare quando ci avviciniamo a meno di una costante $\varepsilon > 0$ prefissata. Si ottiene quindi una dimostrazione di ε-ottimalità per questo algoritmo *applicato a questo particolare problema* (si noti tuttavia che questo non implica che l'algoritmo converga in tempo finito, dato che le normali proprietà di convergenza stocastica degli algoritmi Multistart sono indipendenti dal valore della funzione obiettivo). L'algoritmo di Multistart è come segue.

1. Sia $\varepsilon > 0$ e $\mathbf{x}^*$ un qualsiasi punto in $\mathbb{R}^{3N}$. Sia $\mathbf{x} = \mathbf{x}^*$.
2. Se $f(\mathbf{x}^*) < \varepsilon$, l'algoritmo termina.
3. Si trova un minimo locale $\mathbf{x}' = (x^1, \ldots, x^N)$ a partire dal punto iniziale $\mathbf{x}$.
4. Se $f(\mathbf{x}') < f(\mathbf{x}^*)$ si aggiorna $\mathbf{x}^* \leftarrow \mathbf{x}'$.
5. Si sceglie un nuovo punto iniziale $\mathbf{x} \in \mathbb{R}^{3N}$ a caso.
6. Si torna al passo 2.

Rimane adesso da implementare il passo 3. Utilizziamo a tale scopo il metodo del gradiente[1], che si tratta senza dubbio del più intuitivo metodo di programmazione nonlineare per problemi senza vincoli (si veda il Riquadro 1 per dettagli sui metodi classici di programmazione lineare senza vincoli). Dato un punto iniziale, l'algoritmo del gradiente produce un minimo locale. L'equazione (2) di aggiornamento per la soluzione all'iterazione successiva è $\mathbf{x} \leftarrow \mathbf{x} - \gamma \nabla f(\mathbf{x})$, e quindi l'algoritmo è come segue:

(a) Sia $\varepsilon' > 0$ e $\mathbf{x}$ un qualsiasi punto in $\mathbb{R}^{3N}$.

(b) Se $||\nabla f(\mathbf{x})|| < \varepsilon'$, l'algoritmo termina con minimo locale $\mathbf{x}' = \mathbf{x}$.

(c) Con $\mathbf{d} = -\nabla f(\mathbf{x})$ si calcola γ come in Eq. (3).

(d) Si aggiorna $\mathbf{x} \leftarrow \mathbf{x} + \gamma \mathbf{d}$.

(e) Si torna al passo 2.

Il metodo del gradiente non viene quasi mai utilizzato in pratica, perché sebbene la dimostrazione di convergenza indichi un ordine di convergenza lineare (si veda il Riquadro 3), il comportamento numerico spesso esibisce il fenomeno dello *zig-zagging* (si veda il Riquadro 4).

Scriviamo in Matlab l'implementazione dell'algoritmo Multistart. Abbiamo bisogno di diversi file.

- `mdgp.m`: calcola il valore della funzione obiettivo relativa all'istanza data al punto $\mathbf{x}$.

[1] Conosciuto in Inglese come *steepest descent*.

Riquadro 1.

In generale, i metodi di programmazione nonlineare per problemi senza vincoli nella forma $\min_{\mathbf{x}} f(\mathbf{x})$ sono dei metodi iterativi in cui viene mantenuta una soluzione $\mathbf{x}$ all'iterazione corrente, e la soluzione $\mathbf{x}'$ all'iterazione successiva viene definita come

$$\mathbf{x}' = \mathbf{x} - \gamma D \nabla f(\mathbf{x}), \tag{2}$$

dove D è una matrice definita positiva. In questo modo, si ha che la retta tra $\mathbf{x}$ e $\mathbf{x}'$ ha direzione $\mathbf{d} = -D\nabla f(\mathbf{x})$. Dato che D è definita positiva, per ogni vettore $\mathbf{v}$ si ha $\mathbf{v}^\top D\mathbf{v} > 0$, e quindi $(\nabla f(\mathbf{x}))^\top D \nabla f(\mathbf{x}) > 0$, da cui $(\nabla f(\mathbf{x}))^\top \mathbf{d} < 0$, e quindi $\mathbf{d}$ è una direzione di diminuzione per il valore della funzione obiettivo. Il metodo così definito converge a un ottimo locale. L'ordine di convergenza dipende dalla scelta della matrice D. Per esempio, nel metodo di Newton si utilizza $D = (\nabla^2 f(\mathbf{x}))^{-1}$. Nel metodo del gradiente si sceglie semplicemente $D = I$, la matrice identità. La scelta del passo (il parametro γ) viene fatta in modo da minimizzare il valore della funzione obiettivo, ovvero

$$\gamma = \min_{s \geq 0} f(\mathbf{x} + s\mathbf{d}). \tag{3}$$

Dato che $\mathbf{x}$ (soluzione all'iterazione corrente) e $\mathbf{d}$ sono vettori noti, il problema (3) è una minimizzazione in una dimensione, che di solito può essere effettuata abbastanza velocemente rispetto al resto dell'algoritmo (o alla peggio, approssimata).

Figura 2: Metodi di ottimizzazione locale nonlineare per problemi senza vincoli.

Riquadro 2.

Si consideri una successione di numeri reali f_k che convergono a f^*, e si assuma che $f_k \neq f^*$ per ogni intero k. L'*ordine di convergenza* della successione è il supremo di tutti gli interi nonnegativi p tali che

$$\lim_{k \to \infty} \frac{|f_{k+1} - f^*|}{|f_k - f^*|^p} = \beta < \infty.$$

Se $p = 1$ e $\beta < 1$, la successione ha un ordine di convergenza lineare. Se $p > 1$ o se $p = 1$ e $\beta = 0$, l'ordine di convergenza è superlineare. Se $p = 2$, la successione ha convergenza quadratica.

Figura 3: Definizione di ordine di convergenza di una successione.

```
% mdgp.m
function ofval = mdgp(x)
  ofval = (-2.32868 + (x(1)-x(4))^2 + (x(2)-x(5))^2 + (x(3)-x(6))^2)^2 + ...
          (-6.20702 + (x(1)-x(7))^2 + (x(2)-x(8))^2 + (x(3)-x(9))^2)^2 + ...
        (-14.727 + (x(1)-x(10))^2 + (x(2)-x(11))^2 + (x(3)-x(12))^2)^2 + ...
          (-2.32868 + (x(4)-x(7))^2 + (x(5)-x(8))^2 + (x(6)-x(9))^2)^2 + ...
       (-6.20702 + (x(4)-x(10))^2 + (x(5)-x(11))^2 + (x(6)-x(12))^2)^2 + ...
       (-2.32868 + (x(7)-x(10))^2 + (x(8)-x(11))^2 + (x(9)-x(12))^2)^2;
end
```

- `linesearch.m`: parametrizza la funzione f per mezzo del parametro λ; da utilizzare nella scelta del passo (3). In pratica, sapendo $\mathbf{x}$ e una direzione di discesa $\mathbf{d}$, calcola

Riquadro 3.

La funzione di Rosenbrock è definita come

$$f(\mathbf{x}) = 100(x_2 - x_1)^2 + (1 - x_1)^2,$$

con $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$, e ha un ottimo globale a $\mathbf{x}^* = (1, 1)$. L'algoritmo del gradiente applicato alla funzione di Rosenbrock dal punto iniziale $\mathbf{x} = (-2, 2)$ con tolleranza $\varepsilon' = 0.001$ converge a $(0.99, 0.99)$ in appena 6 passi. Con altri punti iniziali, tuttavia, la convergenza è molto più lenta. Ad esempio, da $\mathbf{x} = (0, 0)$ e limite sulle iterazioni configurato a 500, l'algoritmo termina alla 500-esima iterazione con una tolleranza d'errore a 0.15502 (molto più alta di quella configurata) e soluzione subottimale $(0.91, 0.91)$. Il problema è dato dal comportamento a zig-zag delle direzioni a iterazioni successive. Nelle figure sotto vediamo il comportamento del metodo del gradiente dal punto iniziale $(0, 0)$ (non converge dopo 100 iterazioni) e dal punto iniziale $(-1, 0)$ (converge dopo 6 iterazioni).

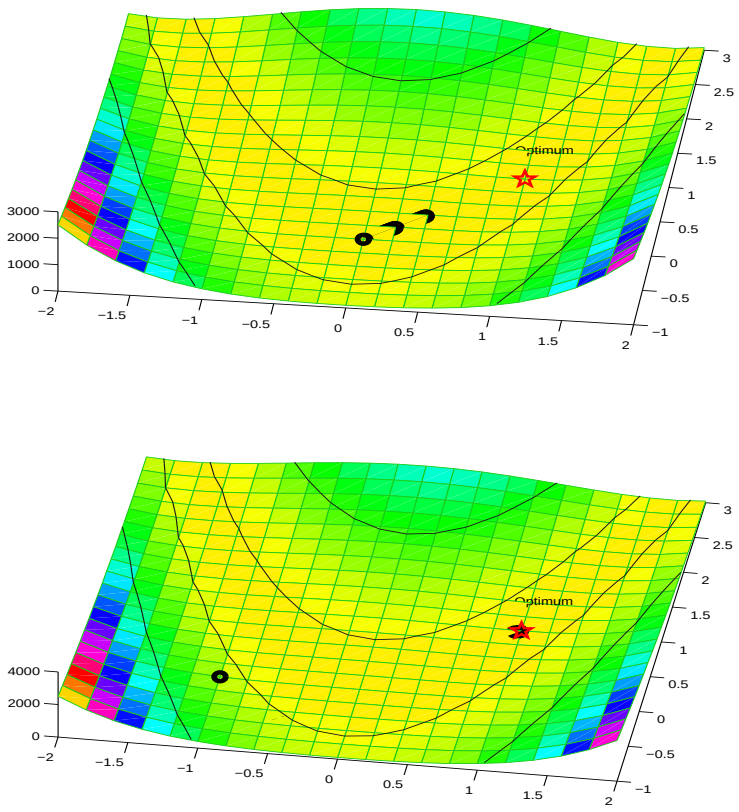

Figura 4: Zig-zagging: algoritmo del gradiente e funzione di Rosenbrock.

$f(\mathbf{x} + \lambda\mathbf{d})$ in funzione di λ.

```
% linesearch.m
function y = linesearch(lambda, f, x, d)
```

```
  y = feval(f, x + lambda*d);
end
```

- `grad.m`: calcola il gradiente $\nabla f(\mathbf{x})$ della funzione f al punto $\mathbf{x}$.

```
% grad.m
function gradf = grad(f, x)
  h = 0.0001;
  n = length(x);
  fval = feval(f, x);
  gradf = zeros(n,1);
  xs = x;
  for i = 1:n
    xs(i) = xs(i) + h;
    gradf(i) = (feval(f, xs) - fval) / h;
    xs(i) = xs(i) - h;
  end
%end function
```

- `steepestdescent.m`: trova un minimo locale $\mathbf{x}^*$ (con valore f^*) della funzione f a partire da un punto iniziale $\mathbf{x}$, dati una tolleranza di subottimalità $\varepsilon' > 0$ e un limite sulle iterazioni. Ritorna anche il numero delle iterazioni effettivamente svolte e la tolleranza effettiva ($||\nabla f(\mathbf{x}^*)||$).

```
% steepestdescent.m
function [xstar, fstar, k, tolerance] = ...
     steepestdescent(f, x, epsilon, maxiterations)
  OPTIONS = [ ];
  termination = 0;
  counter = 1;
  while termination == 0
    d = -grad(f, x);
    tolerance = norm(d, 2);
    if (tolerance < epsilon) | (counter > maxiterations)
      termination = 1;
      xstar = x;
      fstar = feval(f, xstar);
      k = counter;
    else
      lambda = fminbnd('linesearch', 0, 1, OPTIONS, f, x, d);
      x = x + lambda*d;
      counter = counter + 1;
    end
  end
%end function
```

- `rnd.m`: genera un vettore uniformemente casuale di n componenti compresi tra `-bound` e `bound`. Per utilizzare il codice su Octave (anziché Matlab), commentare la riga 3 e decommentare la riga 5.

```
function xrnd = rnd(n, bound)
% instruction when using Matlab
  xrnd = bound * random('Uniform', -ones(n,1), ones(n, 1), n, 1);
```

```
% instruction when using Octave
% xrnd = bound * uniform_rnd(-ones(n,1), ones(n,1));
%end function
```

- `multistart.m`: trova un minimo globale $\mathbf{x}^*$ (con valore f^*) della funzione f (con valore ottimale 0), dati una tolleranza di subottimalità $\varepsilon > 0$ e un limite sulle iterazioni. Ritorna anche il numero delle iterazioni effettivamente svolte. Si noti che si limitano le iterazioni e la tolleranza della minimizzazione locale rispettivamente a 4 e 0.1.

```
% multistart.m
function [xstar, fstar, k] = multistart(f, n, epsilon, maxiterations)
  maxlocalitn = 4;
  localepsilon = 0.1;
  bound = 5;
  x = rnd(n, bound);
  xstar = x;
  counter = 1;
  termination = 0;
  while termination == 0
    fstar = feval(f, xstar);
    if fstar < epsilon | counter > maxiterations
      termination = 1;
      k = counter;
    else
      [xlocal, flocal] = steepestdescent(f, x, localepsilon, maxlocalitn);
      if flocal < fstar
        xstar = xlocal;
        fstar = flocal;
      end
      x = rnd(n, bound);
      counter = counter + 1;
    end
  end
%end function
```

Per lanciare il programma sull'istanza data di 4 atomi, con una tolleranza di 0.1 e un limite massimo di iterazioni di 1000, si utilizza il comando:

```
[xstar, fstar, k] = multistart('mdgp', 12, 0.1, 1000)
```

(il secondo parametro indica il numero di variabili nel problema, che in questo caso è $4 \times 3 = 12$).

Qualche esperimento numerico mostra i limiti della semplicità di questo approccio risolutivo.

- Sia l'algoritmo di soluzione locale sia quello di soluzione globale spesso non convergono per via della tolleranza, bensì per via del numero massimo di iterazioni. In pratica, non è garantita né l'ottimalità locale né quella globale.

- Essendo l'algoritmo multistart un algoritmo stocastico, non si possono replicare i risultati già ottenuti.
- Tarare i parametri della ricerca locale può essere meno utile rispetto a quelli della ricerca globale: in un approccio di questo tipo, con una fase di ricerca globale e una fase di ricerca locale, di solito si ottengono migliori risultati investendo tempo e risorse nella fase globale. Se la fase globale è ben strutturata, ricerche anche approssimative nella fase locale possono portare comunque a una buona approssimazione dell'ottimo.

L'approssimazione migliore, ottenuta su circa 10 tentativi eseguiti su Octave[2], è la seguente:

$$\begin{aligned}\mathbf{x}^* &= (-2.44376, 0.74506, -2.03324, -1.53557, 0.41738, -0.83508,\\ &\quad -1.42520, -1.15822, -0.80798, -0.18048, -1.38518, 0.18503)\end{aligned}$$

con valore della funzione obiettivo $f^* = 0.12$, mostrata in Figura 5.

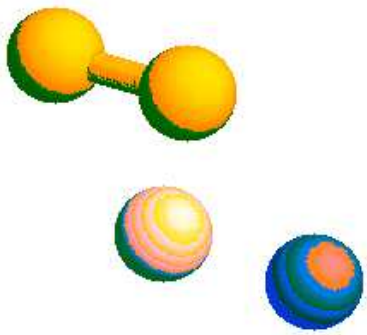

Figura 5: Soluzione sub-ottimale.

[2]Con Matlab sono stati ottenuti risultati numerici migliori; questo comportamento è imputabile a una migliore implementazione della funzione di libreria `fminbnd` usata nella ricerca del passo.