

# Ottimizzazione Non Lineare

G. Liuzzi<sup>1</sup>

Venerdì 24 Novembre 2017

---

<sup>1</sup>Istituto di Analisi dei Sistemi ed Informatica IASI - CNR

# Algoritmo di campionamento uniforme

Se  $\mathcal{A}$  è un intorno di un minimo globale  $x^*$  di  $f$  su  $\mathcal{D}$ , e si generano molti punti a caso su  $\mathcal{D}$ , allora è sempre più probabile che uno di questi punti cada in  $\mathcal{A}$ .

Questa considerazione è alla base del primo metodo di ottimizzazione globale che vediamo.

INPUT:  $\text{maxit} > 0$

Genera  $x_0 \in \mathcal{U}(\mathcal{D})$  e poni  $f_{best} \leftarrow f(x_0)$  e  $x_0^* \leftarrow x_0$

**for**  $k = 1, \dots, \text{maxit}$

Genera  $x_k \in \mathcal{U}(\mathcal{D})$

**if**  $f(x_k) < f_{best}$  **then**  $f_{best} \leftarrow f(x_k)$ ,  $x_k^* \leftarrow x_k$

**else** poni  $x_k^* \leftarrow x_{k-1}^*$

**end for**

OUTPUT:  $\{x_k^*\}$

# Generazione di punti a caso su $\mathcal{D}$

Il risultato fondamentale sul quale si basano molti metodi probabilistici è il seguente.

## Proposizione

*Sia  $\{x_k\}$  una succ. di punti aleatori scelti a caso su  $\mathcal{D}$ . Allora, per ogni sottoinsieme  $\mathcal{A} \subset \mathcal{D}$  tale che*

$$\text{meas}(\mathcal{D}) > \text{meas}(\mathcal{A}) > 0$$

*si ha*

$$\lim_{k \rightarrow \infty} \text{Prob} \{X_k \cap \mathcal{A} \neq \emptyset\} = 1 \quad \text{con } X_k = \{x_0, x_1, \dots, x_k\}.$$

**dim.** Sia

$$p = \frac{\text{meas}(\mathcal{A})}{\text{meas}(\mathcal{D})}$$

per cui risulta  $0 < p < 1$ . È facile riconoscere in  $p$  la probabilità che un punto scelto a caso in  $\mathcal{D}$  appartenga ad  $\mathcal{A}$ .

Generazione di punti a caso su  $\mathcal{D}$ 

Ora, la probabilità che, generati  $k$  vettori, almeno uno appartenga ad  $\mathcal{A}$  è

$$\text{Prob}\{X_k \cap \mathcal{A} \neq \emptyset\} = 1 - \text{Prob}\{X_k \cap \mathcal{A} = \emptyset\} = 1 - (1 - p)^k$$

da questo segue

$$\lim_{k \rightarrow \infty} \text{Prob}\{X_k \cap \mathcal{A} \neq \emptyset\} = 1.$$



## Convergenza dell' Algoritmo di campionamento uniforme

## Proposizione

Sia  $\max_{\mathcal{D}} f = +\infty$  e  $\{x_k^*\}$  la successione (infinita) di punti prodotti dall'algoritmo. Allora, se  $f \in C$ , per ogni  $\epsilon > 0$  si ha:

$$\lim_{k \rightarrow \infty} \text{Prob} \{x_k^* \in \{x \in \mathcal{D} : f(x) < f^* + \epsilon\}\} = 1,$$

con  $f^*$  valore ottimo globale di  $f$  su  $\mathcal{D}$ .

**dim.** Definiamo

$$\mathcal{A} = \{x \in \mathcal{D} : f(x) < f^* + \epsilon\}.$$

La continuità di  $f$  implica che  $\text{meas}(\mathcal{A}) > 0$  e si nota facilmente che

$$\text{Prob} \{x_k^* \in \mathcal{A}\} = \text{Prob} \{X_k \cap \mathcal{A} \neq \emptyset\},$$

dove  $X_k = \{x_0, x_1, \dots, x_k\}$ .

□

# Nota Bene

Quante iterazioni  $k$  occorre eseguire affinché  $Prob\{x_k^* \in \mathcal{A}\} > \alpha \in (0, 1)$ ?

$$Prob\{x_k^* \in \mathcal{A}\} = 1 - \left(1 - \frac{meas(\mathcal{A})}{meas(\mathcal{D})}\right)^k > \alpha$$

$$1 - \alpha > \left(1 - \frac{meas(\mathcal{A})}{meas(\mathcal{D})}\right)^k$$

$$\log(1 - \alpha) > k \log\left(1 - \frac{meas(\mathcal{A})}{meas(\mathcal{D})}\right)$$

$$\frac{\log(1 - \alpha)}{\log\left(1 - \frac{meas(\mathcal{A})}{meas(\mathcal{D})}\right)} < k$$

# Nota Bene

Quante iterazioni  $k$  occorre eseguire affinché  $Prob\{x_k^* \in \mathcal{A}\} > \alpha \in (0, 1)$ ?

$$k > \frac{\log(1 - \alpha)}{\log\left(1 - \frac{meas(\mathcal{A})}{meas(\mathcal{D})}\right)}$$

Se ipotizziamo  $\mathcal{D}$  t.c.  $meas(\mathcal{D}) = 1$  e  $\mathcal{A}$  un ipercubo di lato  $\ell$  quindi  $meas(\mathcal{A}) = \ell^n$ ,

$$k > \frac{\log(1 - \alpha)}{\log(1 - \ell^n)} = \mathcal{O}\left(\frac{1}{\ell^n}\right)!!!$$

# Algoritmo Controlled Random Search (CRS)

Algoritmo in due fasi

- 1 Fase Globale: generazione random di  $m$  punti su  $\mathcal{D}$
- 2 Fase Locale: miglioramento iterativo degli  $m$  punti



## Algoritmo Controlled Random Search (CRS)

INPUT:  $m \gg n$ , maxit, tol

(fase globale) Genera  $S_0 = \{x_i \in \mathcal{U}(\mathcal{D}), i = 1, \dots, m\}$

$k \leftarrow 0$

Calcola  $f_{\min} = \min_{x_i \in S_0} f(x_i)$ ,  $f_{\max} = \max_{x_i \in S_0} f(x_i)$

**while**  $k \leq \text{maxit}$  **and**  $f_{\max} - f_{\min} > \text{tol}$

    Seleziona a caso  $X_k \subset S_k$  t.c.  $|X_k| = n + 1$

    Seleziona a caso  $y \in X_k$

    Calcola  $\bar{x} \leftarrow \sum_{x \in X_k, x \neq y} x / n$

    Calcola  $z \leftarrow 2\bar{x} - y$

**if**  $z \in \mathcal{D}$  **and**  $f(z) < f_{\max}$

**then**  $S_{k+1} \leftarrow S_k \setminus \{x_{\max}\} \cup \{z\}$

**else**  $S_{k+1} \leftarrow S_k$

$k \leftarrow k + 1$

**end**

# Notazione - minimizzazione locale

Sia  $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}^n$  una mappa (che rappresenta un generico metodo di ottimizzazione locale) tale che:

- 1 per ogni  $x \in \mathcal{D}$ ,  $\bar{x} = \mathcal{L}(x)$  è un punto stazionario di  $f$  su  $\mathcal{D}$  tale che  $f(\bar{x}) \leq f(x)$ ;
- 2 per ogni minimo globale  $x^*$ , esiste un aperto  $\mathcal{A}$  tale che se  $x \in \mathcal{A}$ ,  $x^* = \mathcal{L}(x)$ .

# Algoritmo di campionamento uniforme “best start” “multi start”

INPUT:  $\text{maxit} > 0$

Genera  $x_0 \in \mathcal{D}$  e poni  $f_{best} \leftarrow f(x_0)$  e  $x_0^* \leftarrow x_0$   $x_0^* \leftarrow \mathcal{L}(x_0)$ ,  
 $f_{best} \leftarrow f(x_0^*)$

**for**  $k = 0, \dots, \text{maxit}$

Genera un punto  $x_k \in \mathcal{D}$ , poni  $\tilde{x}_k \leftarrow \mathcal{L}(x_k)$

**if**  $f(x_k, \tilde{x}_k) < f_{best}$  **then**  $f_{best} \leftarrow f(x_k)f(\tilde{x}_k)$ ,  $x_k^* \leftarrow x_k\tilde{x}_k$

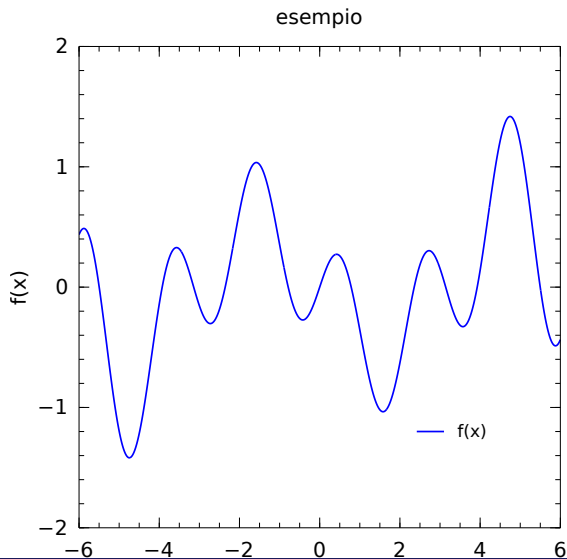
**else** poni  $x_k^* \leftarrow x_{k-1}^*$

**end for**

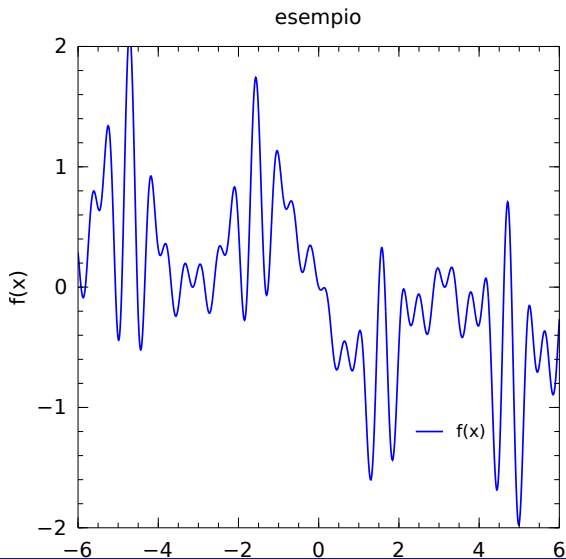
poni  $x_{k+1}^* \leftarrow \mathcal{L}(x_k^*)$ ,  $f_{best} \leftarrow f(x_{k+1}^*)$

OUTPUT:  $\{x_k^*\}$ ,  $f_{best}$

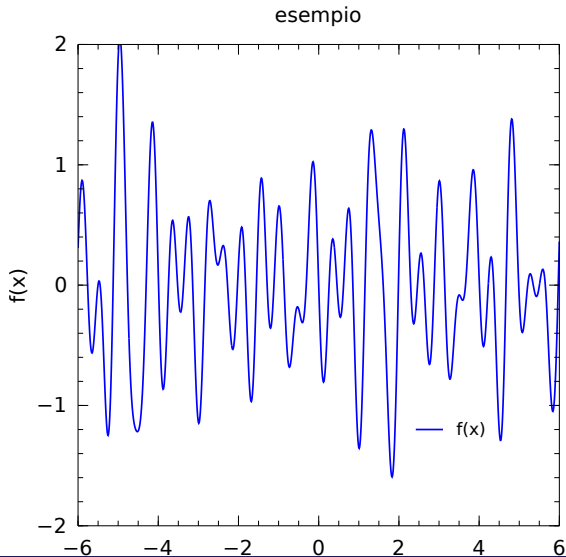
# Funzione obiettivo modificata



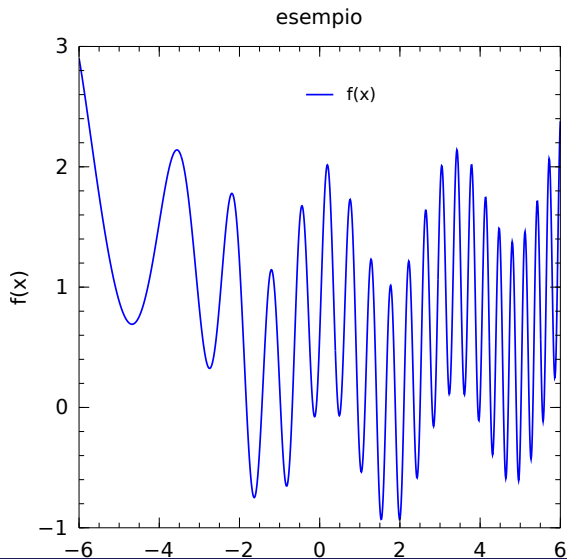
# Funzione obiettivo modificata



# Funzione obiettivo modificata



# Funzione obiettivo modificata



## Algoritmo di tipo “basin hopping”

È una variante “efficiente” del metodo multi-start.

L’idea è quella di migliorare volta per volta il minimo locale corrente generando il punto di partenza  $x_k$  in un intorno  $\mathcal{N}(x_k^*)$  anziché su tutto il dominio  $\mathcal{D}$ .



# Algoritmo “multi start” “basin hopping”

INPUT:  $\text{maxit} > 0$

Genera  $x \in \mathcal{D}$  e poni  $x^* \leftarrow \mathcal{L}(x)$ ,  $f_{best} \leftarrow f(x^*)$

**for**  $k = 0, \dots, \text{maxit}$

Genera  $x \in \mathcal{D}$ , poni  $\tilde{x} \leftarrow \mathcal{L}(x)$

**if**  $f(\tilde{x}) < f_{best}$  **then**  $f_{best} \leftarrow f(\tilde{x})$ ,  $x^* \leftarrow \tilde{x}$

**else** *migliora localmente  $\tilde{x}$*

**while not** criterio di arresto

Genera  $x \in \mathcal{N}(\tilde{x})$ , poni  $y \leftarrow \mathcal{L}(x)$

**if**  $f(y) < f(\tilde{x})$  **then**  $\tilde{x} \leftarrow y$

**end**

**if**  $f(\tilde{x}) < f_{best}$  **then**  $f_{best} \leftarrow f(\tilde{x})$ ,  $x^* \leftarrow \tilde{x}$

**end**

**end for**

OUTPUT:  $x^*$ ,  $f_{best}$

## Algoritmo basin hopping monotono

INPUT:  $\text{maxit} > 0$

Genera  $x \in \mathcal{D}$  e poni  $x^* \leftarrow \mathcal{L}(x)$ ,  $f_{best} \leftarrow f(x^*)$

**for**  $k = 0, \dots, \text{maxit}$  **while not** criterio di arresto globale

Genera  $x \in \mathcal{D}$ , poni  $\tilde{x} \leftarrow \mathcal{L}(x)$

*migliora localmente  $\tilde{x}$*

**while not** criterio di arresto **while not** criterio di arresto locale

Genera  $x \in \mathcal{N}(\tilde{x})$ , poni  $y \leftarrow \mathcal{L}(x)$

**if**  $f(y) < f(\tilde{x})$   *$f(y) < f(\tilde{x})$*  **then**  $\tilde{x} \leftarrow y$

**end**

**if**  $f(\tilde{x}) < f_{best}$  **then**  $f_{best} \leftarrow f(\tilde{x})$ ,  $x^* \leftarrow \tilde{x}$

**end**

**end for**

OUTPUT:  $x^*$ ,  $f_{best}$

## Algoritmo basin hopping nonmonotono

Si sostituisce il criterio

$$f(y) < f(\tilde{x})$$

con il criterio

$$\mathcal{U}(0, 1) \leq \exp(-(f(y) - f(\tilde{x}))/T)$$

- quando  $f(y) < f(\tilde{x})$ ,  $\exp(-(f(y) - f(\tilde{x}))/T) > 1$  quindi il criterio è banalmente soddisfatto
- quando  $f(y) \geq f(\tilde{x})$ ,  $\exp(-(f(y) - f(\tilde{x}))/T) \leq 1$  quindi
  - se  $\mathcal{U}(0, 1) > \exp(-(f(y) - f(\tilde{x}))/T)$  il punto proposto è “rifiutato”
  - se  $\mathcal{U}(0, 1) \leq \exp(-(f(y) - f(\tilde{x}))/T)$  il punto proposto è “accettato” anche se è un punto peggiore rispetto a  $\tilde{x}$

# Algoritmo basin hopping nonmonotono

INPUT:  $\text{maxit} > 0$

Genera  $x \in \mathcal{D}$  e poni  $x^* \leftarrow \mathcal{L}(x)$ ,  $f_{best} \leftarrow f(x^*)$

**while not** criterio di arresto globale

Genera  $x \in \mathcal{D}$ , poni  $\tilde{x} \leftarrow \mathcal{L}(x)$

*migliora localmente  $\tilde{x}$*

**while not** criterio di arresto locale

Genera  $x \in \mathcal{N}(\tilde{x})$ , poni  $y \leftarrow \mathcal{L}(x)$

**if**  $\mathcal{U}(0, 1) \leq \exp(-(f(y) - f(\tilde{x}))/T)$  **then**  $\tilde{x} \leftarrow y$

**end**

**if**  $f(\tilde{x}) < f_{best}$  **then**  $f_{best} \leftarrow f(\tilde{x})$ ,  $x^* \leftarrow \tilde{x}$

**end**

**end for**

OUTPUT:  $x^*$ ,  $f_{best}$

# Basin hopping – crit. di arresto locale

Un criterio molto usato consiste nell'imporre un massimo numero di iterazioni in cui non si aggiorna  $\tilde{x}$

Il ciclo interno diventa:

$h \leftarrow 0$

**while**  $h \leq \text{max\_no\_improve}$

Genera  $x \in \mathcal{N}(\tilde{x})$ , poni  $y \leftarrow \mathcal{L}(x)$

**if**  $y \prec x$  **then**  $\tilde{x} \leftarrow y$ ,  $h \leftarrow 0$

**else**  $h \leftarrow h + 1$

**end**