

Sull'algoritmo DIRECT e le sue evoluzioni

Gian Maria Negri Porzio

Luglio 2016



Piano della presentazione

- ① Concetti base
- ② L'algoritmo di Shubert
- ③ L'algoritmo DIRECT
- ④ La modifica di Finkel e Kelley



Il nostro scopo

Spesso nelle applicazioni reali non abbiamo a disposizione informazioni sulle derivate:

- Costo computazionale eccessivo.
- Regolarità insufficiente.



Il nostro scopo

Spesso nelle applicazioni reali non abbiamo a disposizione informazioni sulle derivate:

- Costo computazionale eccessivo.
- Regolarità insufficiente.

È stato dunque necessario sviluppare metodi per calcolare minimi che **non** richiedano forti **ipotesi di regolarità**.



Il problema

Definizione

Un sottoinsieme $R \subset \mathbb{R}^n$ si dice *iperrettangolo (chiuso)* se può essere scritto nella forma

$$R = \bigotimes_{i=1}^n [a_i, b_i],$$

con $a_i, b_i \in \mathbb{R}$.



Il problema

Definizione

Un sottoinsieme $R \subset \mathbb{R}^n$ si dice *iperrettangolo (chiuso)* se può essere scritto nella forma

$$R = \bigotimes_{i=1}^n [a_i, b_i],$$

con $a_i, b_i \in \mathbb{R}$.

Problema

Data $f : R \rightarrow \mathbb{R}$ funzione K -lipschitziana da un iperrettangolo $R \subset \mathbb{R}^n$ a \mathbb{R} , vogliamo \bar{x} tale che

$$f(\bar{x}) \leq f(y) \quad \forall y \in R,$$



Il problema

Le condizioni per gli algoritmi che presenteremo sono le seguenti:

- Usare solamente valutazioni della funzione f ed evitare eventuali informazioni differenziali.
- Se possibile, non utilizzare esplicitamente la costante di Lipschitz K .



L'antenato di DIRECT

Shubert presentò una sua soluzione nel 1972. Consideriamo $f : [a, b] \rightarrow \mathbb{R}$ K -lipschitziana. Dalla definizione di lipschitzianità seguono facilmente le disuguaglianze:

$$\begin{aligned}f(x) &\geq f(a) - K(x - a), \\f(x) &\geq f(b) + K(x - b).\end{aligned}$$



L'idea di Shubert

Dato un intervallo $[a_i, b_i]$, definiamo:

$$X(a_i, b_i, f, K) = \frac{a_i + b_i}{2} + \frac{f(a_i) - f(b_i)}{2K},$$

$$B(a_i, b_i, f, K) = \frac{f(a_i) + f(b_i)}{2} - K(b_i - a_i).$$



L'idea di Shubert

Dato un intervallo $[a_i, b_i]$, definiamo:

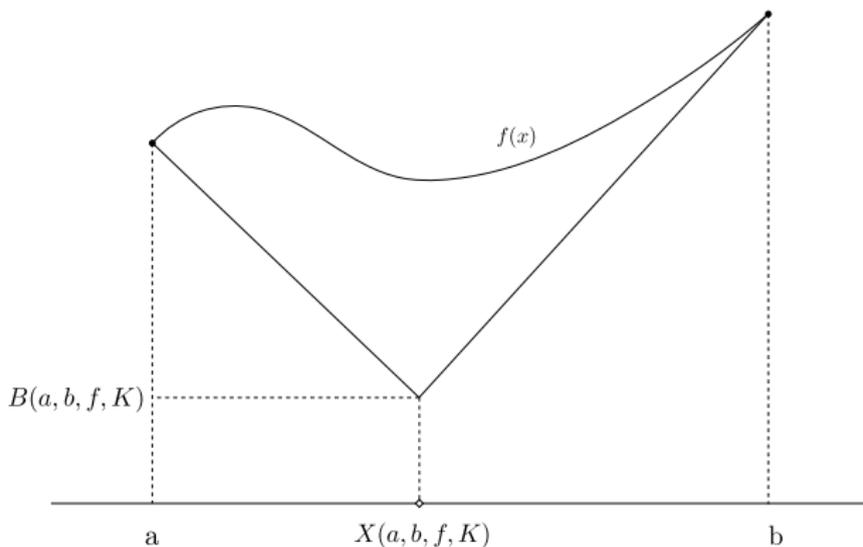
$$X(a_i, b_i, f, K) = \frac{a_i + b_i}{2} + \frac{f(a_i) - f(b_i)}{2K},$$
$$B(a_i, b_i, f, K) = \frac{f(a_i) + f(b_i)}{2} - K(b_i - a_i).$$

Queste due quantità corrispondono alla ascissa e all'ordinata del minimo valore che può assumere la nostra funzione nell'intervallo $[a_i, b_i]$.



L'idea di Shubert

Figura: Rappresentazione grafica di $X(a, b, f, K)$ e $B(a, b, f, K)$.



L'idea di Shubert

L'idea di Shubert è approssimare la funzione f dal basso con una funzione lineare a tratti il cui modulo della derivata è dato da K .



L'idea di Shubert

L'idea di Shubert è approssimare la funzione f dal basso con una funzione lineare a tratti il cui modulo della derivata è dato da K . Vediamone lo pseudo codice e i primi passi.



Lo pseudo codice

Data f K -lipschitziana definita su $[a, b]$:

```

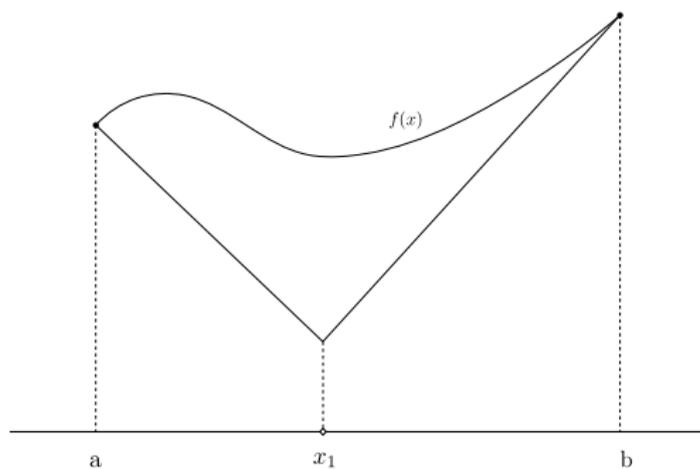
function Fmin = shubert(f, a, b, K)
  Fmin = min{f(a), f(b)};
  Inter = {[a, b]};
  while termine
    [l, u] = min_B(f, K){Inter};
    x = X(l, u, f, K);
    update(Fmin); Inter.del([l, u]);
    Inter.add([l, x]); Inter.add([x, u]);
    check(termine);
  end
  return Fmin
end

```



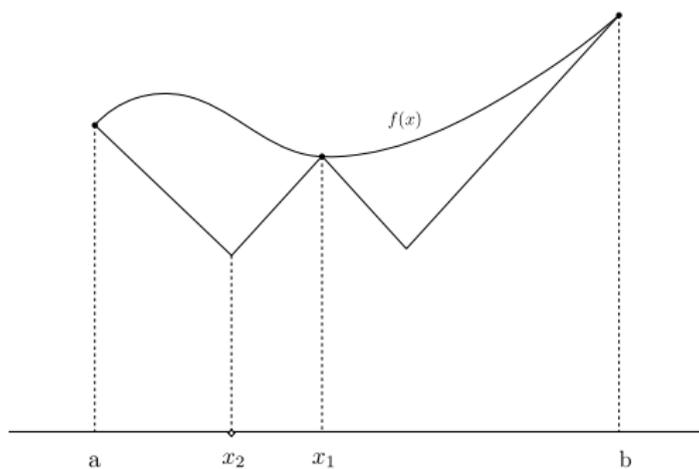
I primi passi di Shubert

Figura: Calcolo di $X(a, b, f, K)$



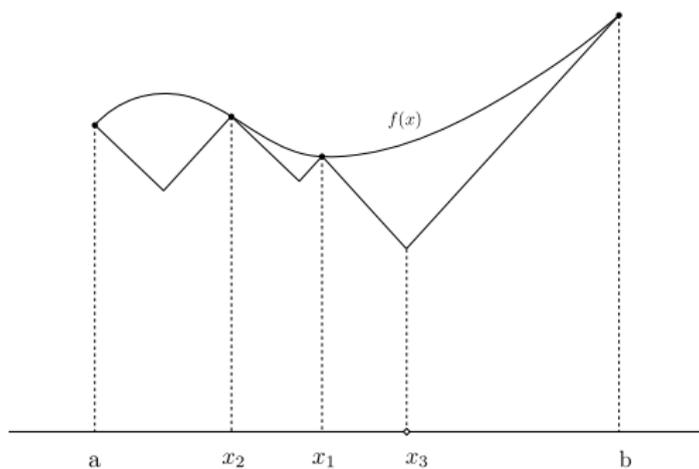
I primi passi di Shubert

Figura: Seconda iterazione.



I primi passi di Shubert

Figura: Scelta del minimo x_3 .



Gli svantaggi di Shubert

Oggigiorno l'algoritmo di Shubert non viene più utilizzato, in quanto presenta tre grandi svantaggi:

- Costo esponenziale in funzione della dimensione.



Gli svantaggi di Shubert

Oggigiorno l'algoritmo di Shubert non viene più utilizzato, in quanto presenta tre grandi svantaggi:

- Costo esponenziale in funzione della dimensione.
- Utilizzo esplicito della costante di Lipschitz o di una sua stima.



Gli svantaggi di Shubert

Oggigiorno l'algorithmo di Shubert non viene più utilizzato, in quanto presenta tre grandi svantaggi:

- Costo esponenziale in funzione della dimensione.
- Utilizzo esplicito della costante di Lipschitz o di una sua stima.
- Convergenza lenta nelle applicazioni reali.



L'algoritmo DIRECT

Nel 1993 Jones, Perttunen e Stuckman presentarono un'alternativa al agli algoritmi di quegli anni:

DIRECT nasce come risposta ai punti deboli di Shubert e ancora oggi è il nucleo di molti software di minimizzazione.



I vantaggi di DIRECT

L'algorithmo DIRECT

- Non necessita della costante di Lipschitz.



I vantaggi di DIRECT

L'algoritmo DIRECT

- Non necessita della costante di Lipschitz.
- Non soffre della *maledizione della dimensionalità*.



I vantaggi di DIRECT

L'algoritmo DIRECT

- Non necessita della costante di Lipschitz.
- Non soffre della *maledizione della dimensionalità*.
- Acronimo di *Dividing RECTangles*.



I vantaggi di DIRECT

L'algoritmo DIRECT

- Non necessita della costante di Lipschitz.
- Non soffre della *maledizione della dimensionalità*.
- Acronimo di *Dividing RECTangles*.



I vantaggi di DIRECT

L'algoritmo DIRECT

- Non necessita della costante di Lipschitz.
- Non soffre della *maledizione della dimensionalità*.
- Acronimo di *D*ividing *R*ECTangles.

Vediamolo in primo luogo in una dimensione. Supporremo per ora di conoscere la costante K .



L'idea di base in una dimensione

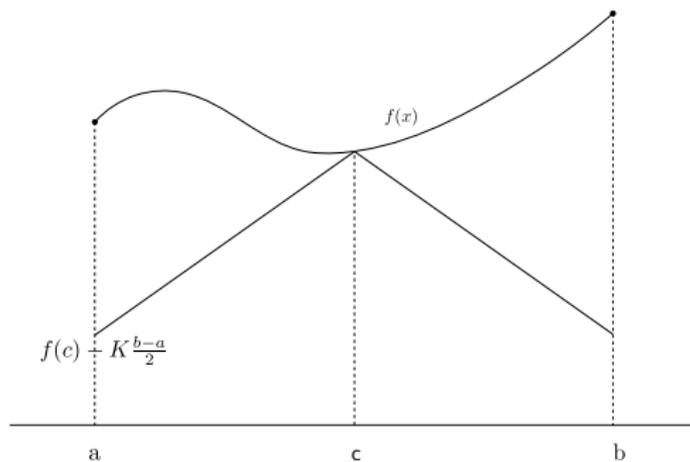
Indicato con $c = \frac{a+b}{2}$ il punto medio dell'intervallo, valgono le stime:

$$f(x) \geq f(c) + K(x - c) \quad \forall x \leq c,$$

$$f(x) \geq f(c) - K(x - c) \quad \forall x \geq c.$$

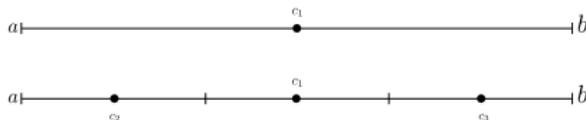


L'idea di base in una dimensione



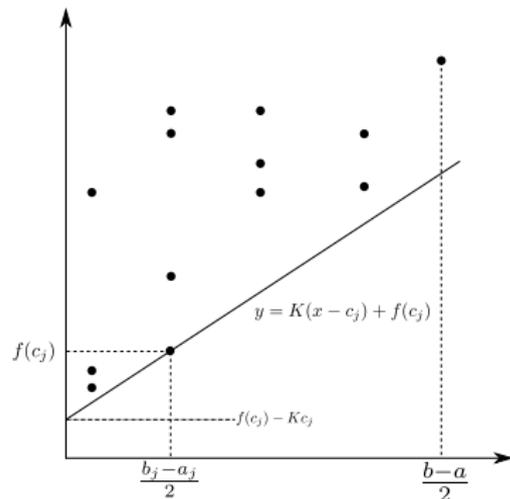
Suddivisione dell'intervallo

Come nell'algoritmo di Shubert, bisogna decidere come quali nuovi punti campionare e come suddividere l'intervallo. L'idea è dividere in terzi e mantenere la centralità.



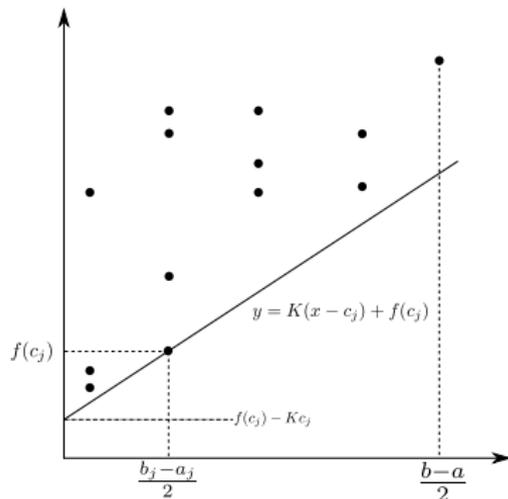
Selezione degli intervalli, K nota

- 1 Rappresentazione bidimensionale degli intervalli.



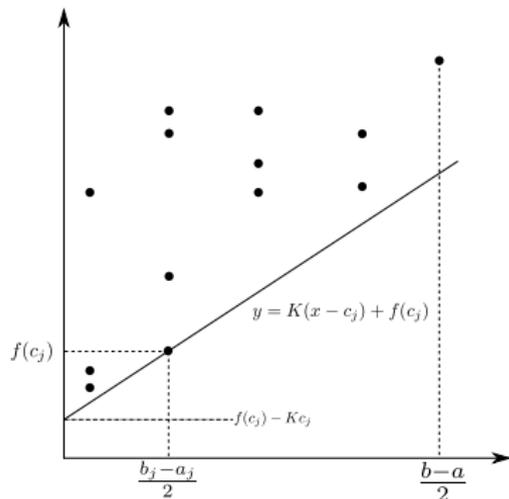
Selezione degli intervalli, K nota

- ① Rappresentazione bidimensionale degli intervalli.
- ② Selezione del miglior intervallo.



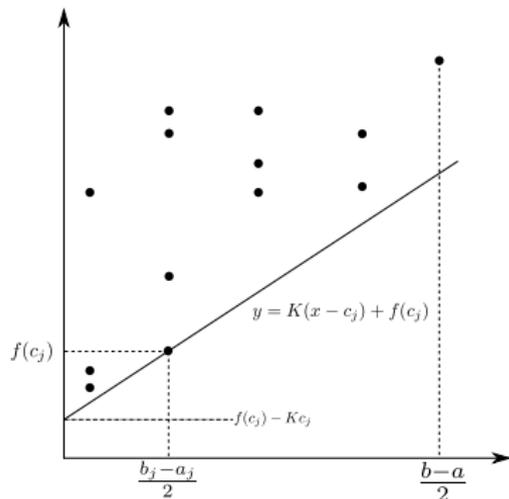
Selezione degli intervalli, K nota

- ① Rappresentazione bidimensionale degli intervalli.
- ② Selezione del miglior intervallo.
- ③ Considerazioni su ricerca globale locale.



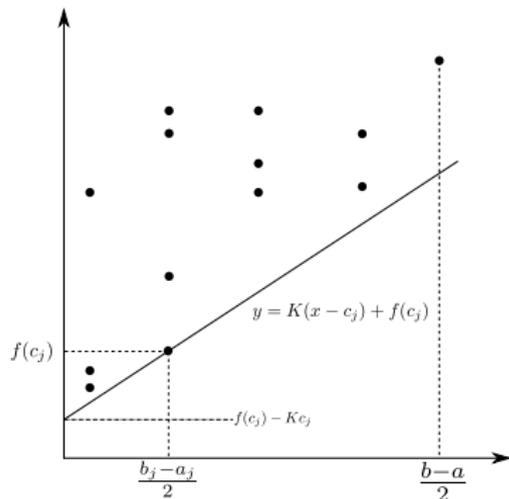
Selezione degli intervalli, K nota

- ① Rappresentazione bidimensionale degli intervalli.
- ② Selezione del miglior intervallo.
- ③ Considerazioni su ricerca globale locale.



Selezione degli intervalli, K nota

- ① Rappresentazione bidimensionale degli intervalli.
- ② Selezione del miglior intervallo.
- ③ Considerazioni su ricerca globale locale.

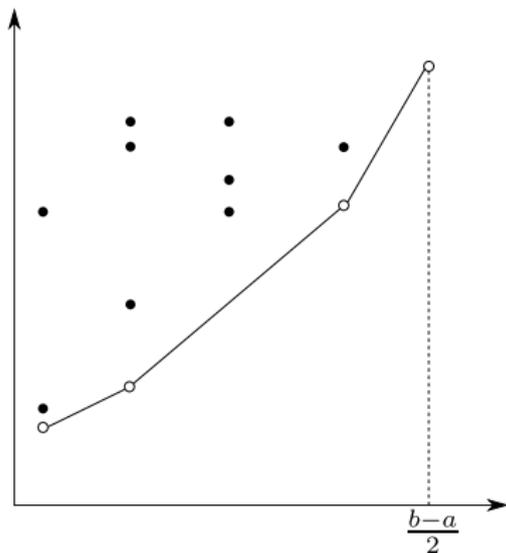


E se non avessimo la costante K ?



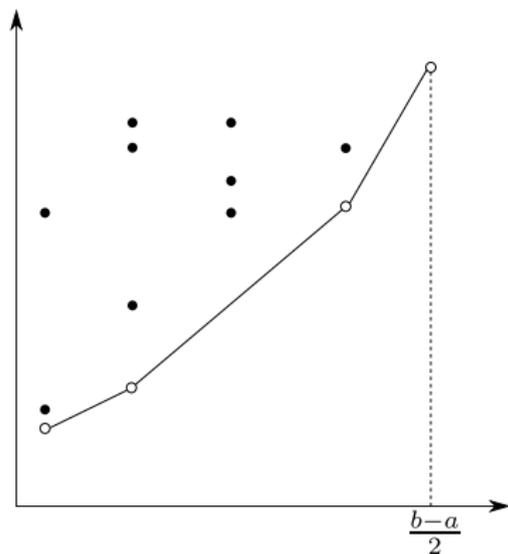
Selezione degli intervalli, K ignota

- 1 Testare ogni possibile \tilde{K} .



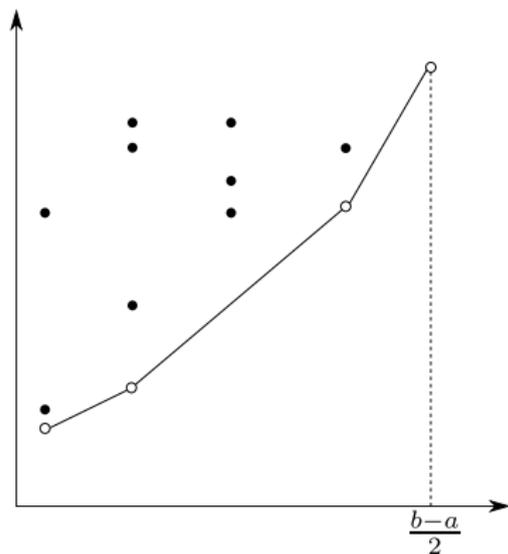
Selezione degli intervalli, K ignota

- 1 Testare ogni possibile \tilde{K} .
- 2 Calcolare la parte inferiore dell'involuppo convesso.



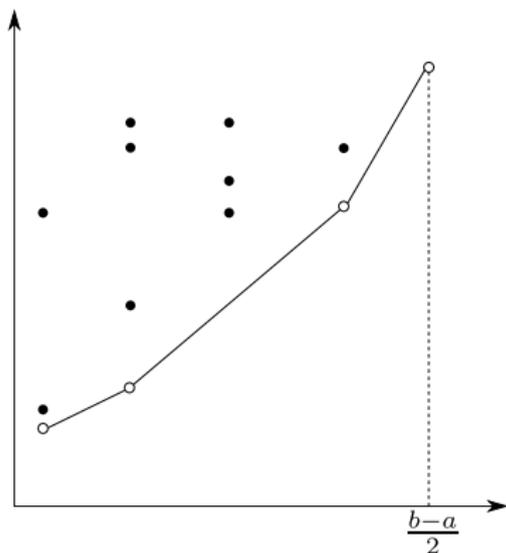
Selezione degli intervalli, K ignota

- 1 Testare ogni possibile \tilde{K} .
- 2 Calcolare la parte inferiore dell'involuppo convesso.



Selezione degli intervalli, K ignota

- 1 Testare ogni possibile \tilde{K} .
- 2 Calcolare la parte inferiore dell'involuppo convesso.



Un possibile metodo è lo scan di Graham.



Iperrettangoli potenzialmente ottimi



Iperrettangoli potenzialmente ottimi

Definizione (Iperrettangoli potenzialmente ottimi)

Sia R un iperrettangolo in \mathbb{R}^n partizionato in R_i per $i = 1, \dots, m$ di centri c_i . Sia f lipschitziana definita su R a valori reali e sia f_{\min} il valore minimo finora computato. Diciamo che R_j è un *iperrettangolo potenzialmente ottimo* se esiste $\tilde{K} > 0$ tale che:

$$f(c_j) - \tilde{K}c_j \leq f(c_i) - \tilde{K}c_i \quad \forall i = 1, \dots, m,$$

$$f(c_j) - \tilde{K}c_j \leq f_{\min} - \varepsilon|f_{\min}|,$$

dove $\varepsilon > 0$ è una costante definita a priori.



Il ruolo di ε

La costante ε serve solo da un punto di vista computazionale: non vogliamo che considerare iperrettangoli se questi non migliorano significativamente il valore minimo.



Il ruolo di ε

La costante ε serve solo da un punto di vista computazionale: non vogliamo che considerare iperrettangoli se questi non migliorano significativamente il valore minimo.

Gli autori consigliano il valore $\varepsilon = 10^{-4}$.



Pseudocodice per 1-DIRECT

Data f lipschitziana definita su $[a, b]$:

```
function Fmin = direct(f, a, b, maxit)
    Int = {[a, b]}
    Fmin = f( (a+b)/2 )
    for it =1:maxit
        S = optInt(Int)
        for j in S
            divide(j)
            update(Fmin)
            update(Int)
        endfor
    endfor
    return Fmin
end
```



DIRECT in più dimensioni

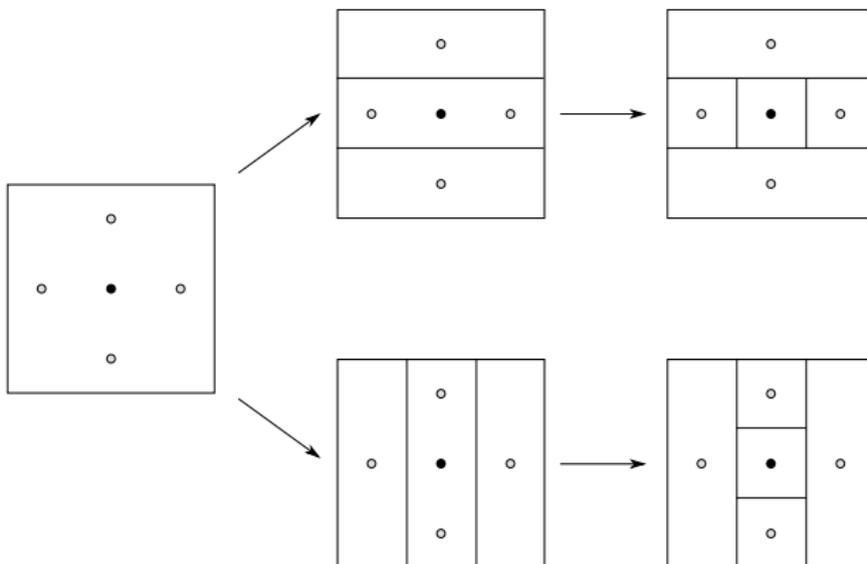
La differenza del caso multidimensionale consiste nello scegliere come suddividere i rettangoli.

È infatti possibile scegliere diversi per le suddivisioni.



Ordine delle suddivisioni

Figura: Le due possibili suddivisioni in \mathbb{R}^2



Ordine delle suddivisioni

Alcune osservazioni

- Nel caso di rettangoli, è *necessario* dividere solo lungo le dimensioni con lunghezza maggiore.



Ordine delle suddivisioni

Alcune osservazioni

- Nel caso di rettangoli, è *necessario* dividere solo lungo le dimensioni con lunghezza maggiore.
- Sperimentalmente i risultati migliori si ottengono quando la funzione ha valori minori sui rettangoli più grandi.



Ordine delle suddivisioni

Alcune osservazioni

- Nel caso di rettangoli, è *necessario* dividere solo lungo le dimensioni con lunghezza maggiore.
- Sperimentalmente i risultati migliori si ottengono quando la funzione ha valori minori sui rettangoli più grandi.



Ordine delle suddivisioni

Alcune osservazioni

- Nel caso di rettangoli, è *necessario* dividere solo lungo le dimensioni con lunghezza maggiore.
- Sperimentalmente i risultati migliori si ottengono quando la funzione ha valori minori sui rettangoli più grandi.

Possiamo ora vedere lo pseudo codice di DIRECT.



Pseudo codice di DIRECT

Data f lipschitziana definita su R :

```
function Fmin = direct(f, R, maxit)
    Fmin = f(c)
    Rect = {R}
    for it =1:maxit
        S = optRect(Rect)
        for j in S
            divide(j)
            update(Fmin)
            update(Rect)
        endfor
    endfor
    return Fmin
end
```



I primi passi di DIRECT

Mostriamo i primi passi per la funzione di Branin:

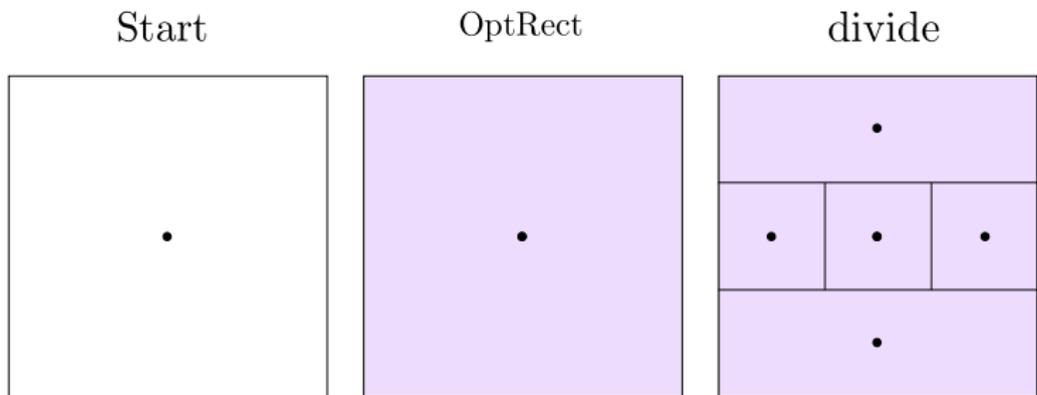
$$\left(y - \frac{5.1}{4\pi^2}x^2 + \frac{5}{\pi}x - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x + 10$$

con $(x, y) \in [-5, 10] \times [0, 15]$.



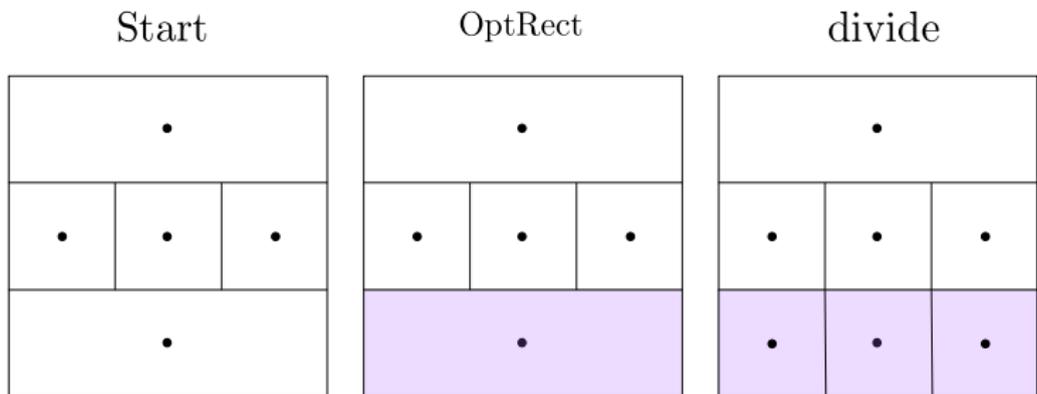
I primi passi di DIRECT

Figura: Prima iterazione



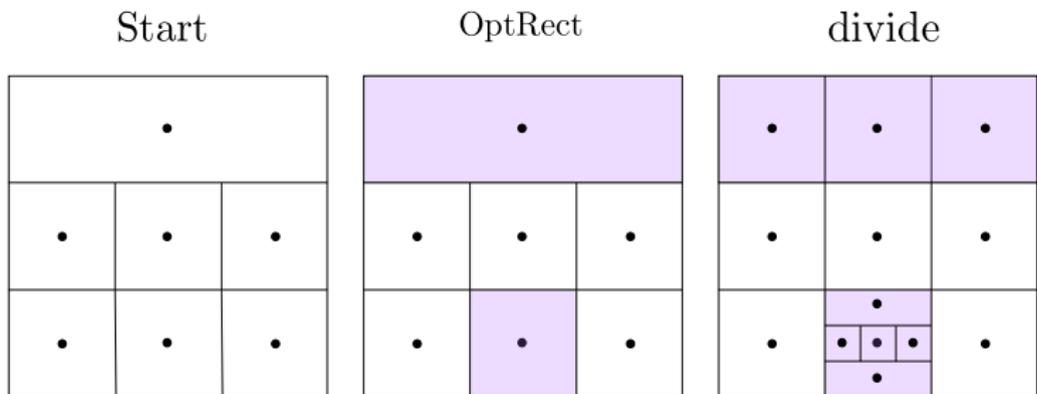
I primi passi di DIRECT

Figura: Seconda iterazione



I primi passi di DIRECT

Figura: Terza iterazione



Convergenza di DIRECT

Teorema (Convergenza dell'algoritmo DIRECT)

Sia $f : R \rightarrow \mathbb{R}$ una funzione Lipschitziana su un iperrettangolo $R \subset \mathbb{R}^n$ a valori reali. Sia (x_j) la successione dei punti dell'algoritmo DIRECT. Allora (x_j) ha limite \bar{x} e

$$f(\bar{x}) = \min_{x \in R} f(x)$$



Alcune modifiche consigliate

Negli anni sono state suggerite diverse modifiche.

- Suddividere al più un solo iperretangolo potenzialmente ottimo.



Alcune modifiche consigliate

Negli anni sono state suggerite diverse modifiche.

- Suddividere al più un solo iperretangolo potenzialmente ottimo.
- Suddividere lungo una sola direzione di lunghezza maggiore.



Alcune modifiche consigliate

Negli anni sono state suggerite diverse modifiche.

- Suddividere al più un solo iperretangolo potenzialmente ottimo.
- Suddividere lungo una sola direzione di lunghezza maggiore.
- Modificare la scelta degli iperretangoli potenzialmente ottimi.



L'additive scaling

Finkel e Kelley si accorsero che in alcune situazioni DIRECT non si comportava benissimo. La velocità di convergenza infatti diminuisce notevolmente nei casi

$$f(x) + M$$

con $M \gg 1$.



Sugli iperrettangoli adiacenti

Teorema

Nelle consuete ipotesi, sia $\{R_i\}$ l'insieme degli iperrettangoli creati da DIRECT, A ipercubo di centro c e lunghezza dei lati 3^{-l} , d_{R_i} distanza del centro dai vertici nel rettangolo R_i . Se

- ① $d_A \leq d_{R_i} \quad \forall i.$
- ② $f(c) = f_{min} \neq 0.$
- ③ $d_A < \frac{\epsilon |f(c)|}{2K} \left(\sqrt{n+8} - \sqrt{n} \right)$

allora A non è iperrettangolo potenzialmente ottimo fintanto che tutti gli iperrettangoli adiacenti avranno la stessa dimensione di A .



Perdita dell'ottimalità

Teorema

Siano $f, c, A, R, \{R_i\}$ come sempre e sia $f^* = \min_{x \in R} f(x)$. Se

$$f^* > \frac{K\sqrt{n}}{\varepsilon(\sqrt{1 + \frac{8}{n}} - 1)}$$

allora A non è potenzialmente ottimo se esiste R_i più grande di A .



Una nuova definizione

Definizione (Iperrettangoli p. o. secondo Finkel e Kelley)

Sia f come nella definizione precedente. Chiamiamo f_{\min} , f_{median} il minimo e la media aritmetica dei valori calcolati da DIRECT fino a quel momento. Diciamo che A è un *iperrettangolo potenzialmente ottimo* secondo F. e K. se esiste \tilde{K} tale che

$$f(c_A) - \tilde{K} \frac{d_A}{2} \leq f(c_T) - \tilde{K} \frac{d_T}{2} \quad \forall T \text{ iperrettangolo}$$

$$f(c_A) - \tilde{K} d_A \leq f_{\min} - \varepsilon |f_{\min} - f_{\text{median}}|$$

dove $\varepsilon > 0$ è una costante definita a priori.



Il software oggi disponibile

Dato il grande successo di DIRECT, oggi giorno sono disponibili numerose implementazioni

- L'ambiente TOMLAB (MULTIMIN, GLCFAST, GLCCLUSTER) per MATLAB.



Il software oggi disponibile

Dato il grande successo di DIRECT, oggi giorno sono disponibili numerose implementazioni

- L'ambiente TOMLAB (MULTIMIN, GLCFAST, GLCCLUSTER) per MATLAB.
- La libreria Dakota per C++.



Il software oggi disponibile

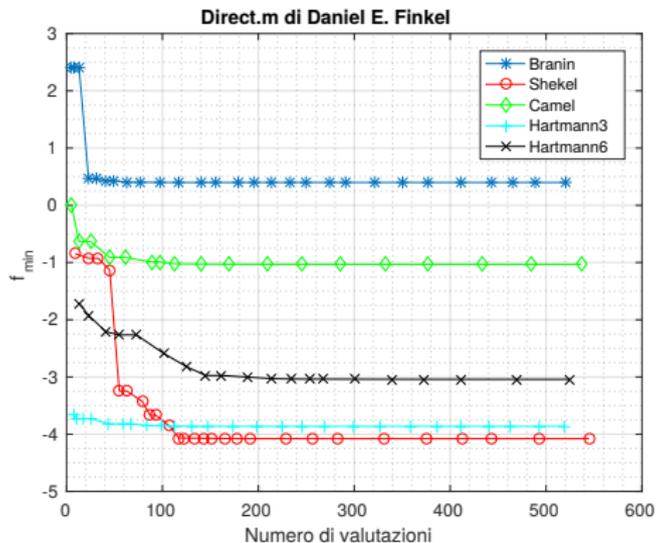
Dato il grande successo di DIRECT, oggigiorno sono disponibili numerose implementazioni

- L'ambiente TOMLAB (MULTIMIN, GLCFAST, GLCCLUSTER) per MATLAB.
- La libreria Dakota per C++.
- Direct.m per MATLAB.



Un esempio numerico.

Figura: Velocità di convergenza di DIRECT.



Considerazioni finali

Riassumendo

- ① Abbiamo presentato l'algoritmo DIRECT seguendo un percorso storico.
- ② Abbiamo visto una sua importante modifica.
- ③ Abbiamo accennato al software oggi disponibile.

