Ottimizzazione Globale

F. Rinaldi

Dipartimento di Matematica
Università di Padova

Corso di Laurea Matematica
2014, Padova

Outline

Ottimizzazione Globale

Formulazione del Problema

Formulazione

Il problema che vogliamo risolvere è il seguente:

$$\min_{x \in S \subseteq R^n} f(x)$$

vogliamo trovare

$$f^\star = \min_{x \in S \subseteq R^n} f(x)$$

e

$$x^\star = \arg \min_{x \in S \subseteq R^n} f(x)$$

Formulazione del Problema

Formulazione

Il problema che vogliamo risolvere è il seguente:

$$\min_{x \in S \subseteq R^n} f(x)$$

vogliamo trovare

$$f^\star = \min_{x \in S \subseteq R^n} f(x)$$

e

$$x^\star = \arg \min_{x \in S \subseteq R^n} f(x)$$

Minimo Locale e Globale

Minimo Globale

$x^\star \in S \subseteq R^n$ è un punto di minimo globale se risulta:

$$f(x^\star) \leq f(x), \quad \forall x \in S.$$

Minimo Locale

$x^\star \in S \subseteq R^n$ è un punto di minimo locale se esiste un intorno $B(x^\star, \varepsilon) \cap S$ tale che:

$$f(x^\star) \leq f(x), \quad \forall x \in B(x^\star, \varepsilon) \cap S.$$

Se funzione f convessa minimo globale e minimo locale coincidono

Minimo Locale e Globale

Minimo Globale

$x^\star \in S \subseteq R^n$ è un punto di minimo globale se risulta:

$$f(x^\star) \leq f(x), \quad \forall x \in S.$$

Minimo Locale

$x^\star \in S \subseteq R^n$ è un punto di minimo locale se esiste un intorno $B(x^\star, \varepsilon) \cap S$ tale che:

$$f(x^\star) \leq f(x), \quad \forall x \in B(x^\star, \varepsilon) \cap S.$$

Se funzione f convessa minimo globale e minimo locale coincidono

Globale vs Locale

- **Ottimizzazione Globale**: trovare un ottimo globale
- **Ottimizzazione Locale**: trovare un ottimo locale
- Ottimo globale necessario (o almeno desiderabile) in molte applicazioni
- Ottimo globale difficile da identificare
- Ottimizzazione convessa: Ottimo locale = Ottimo globale

Applicazioni dell'Ottimizzazione Globale

Numerosi problemi reali

- Ottimizzazione discreta (Packing, Scheduling, . . .)
- Teoria dei grafi
- Minimi quadrati nonlineari
- Biologia e Chimica (e.g. Protein Folding, Problemi di Equilibrio)
- Robotica
- . . .

Ottimizzazione discreta

Possibile dimostrare equivalenza tra questi due problemi:

$$\begin{aligned} &\min c^T x \\ &x \in P \\ &x \in \{0,1\}^n \end{aligned} \tag{1}$$

$$\begin{aligned} &\min c^T x + \frac{1}{\varepsilon} \sum_{i=1}^{n} x_i(1 - x_i) \\ &x \in P \\ &0 \leq x \leq e \end{aligned} \tag{2}$$

con $e \in R^n$ vettore unitario, e $\varepsilon > 0$ parametro di penalità.

[M. Raghavachari, 1969]
[F. Giannessi and F. Niccolucci, 1976]

Problema di Programmazione Concava

Ottimizzazione discreta

Possibile dimostrare equivalenza tra questi due problemi:

$$\begin{aligned} &\min c^T x \\ &x \in P \\ &x \in \{0,1\}^n \end{aligned} \tag{1}$$

$$\begin{aligned} &\min c^T x + \frac{1}{\varepsilon} \sum_{i=1}^{n} x_i(1 - x_i) \\ &x \in P \\ &0 \leq x \leq e \end{aligned} \tag{2}$$

con $e \in R^n$ vettore unitario, e $\varepsilon > 0$ parametro di penalità.

[M. Raghavachari, 1969]
[F. Giannessi and F. Niccolucci, 1976]

Problema di Programmazione Concava

Teoria dei Grafi

Problema della massima clique (ricerca del numero massimo di vertici mutuamente adiacenti):

$$\begin{aligned} &\max\ x^T A x \\ &e^T x = 1 \\ &x \geq 0 \end{aligned} \tag{3}$$

con A matrice di adiacenza del grafo. [Motzkin e Strauss, 1965]

Problema di Programmazione Quadratica Indefinita

Teoria dei Grafi

Problema della massima clique (ricerca del numero massimo di vertici mutuamente adiacenti):

$$\begin{aligned} &\max\ x^T A x \\ &e^T x = 1 \\ &x \geq 0 \end{aligned} \tag{3}$$

con A matrice di adiacenza del grafo. [Motzkin e Strauss, 1965]

Problema di Programmazione Quadratica Indefinita

Minimi Quadrati Non Lineari

Considerato l'insieme

$$T = \{(x^i, y^i),\ x^i \in R^n,\ y^i \in R,\ i = 1, \ldots, L\}$$

Trovare una funzione che approssima i dati (Data Fitting):

$$\min \sum_{i=1}^{L} \|y^i - F(x^i, \theta)\|^2 \tag{4}$$

Con θ vettore di parametri e F funzione non lineare in θ.

Problema di Programmazione Non convessa

Minimi Quadrati Non Lineari

Considerato l'insieme

$$T = \{(x^i, y^i),\ x^i \in R^n,\ y^i \in R,\ i = 1, \ldots, L\}$$

Trovare una funzione che approssima i dati (Data Fitting):

$$\min \sum_{i=1}^{L} \|y^i - F(x^i, \theta)\|^2 \tag{4}$$

Con θ vettore di parametri e F funzione non lineare in θ.

Problema di Programmazione Non convessa

Altri Problemi

Biologia e Chimica

- **Protein Folding**: Trovare la configurazione a minima energia di N atomi in una proteina (con una data sequenza amminoacidica) assumendo che le forze agenti tra gli atomi siano note (Minimizzare la funzione energia potenziale).
- **Problemi di Equilibrio**: Trovare il numero e la composizione delle fasi di una mistura di sostanze chimiche all'equilibrio (Minimizzare la funzione dell'energia libera di Gibbs).

Robotica

- Numerosi problemi di Robotica (Trajectory planning, Sensing, Control, Performances evaluation, Design, Calibration, ...) possono essere formulati come problemi di ottimizzazione globale.

Algoritmi di Ottimizzazione Globale

- Algoritmi che usano informazioni globali (convessità, concavità, costante di Lipschitz della funzione obiettivo e dei vincoli, numero di minimi globali, . . .)
- Algoritmi che usano informazioni locali (valori della funzione obiettivo e dei vincoli nei vari punti , valori delle derivate,. . .)

Proprietà di convergenza ottenibili soltanto attraverso un campionamento opportuno

- **Algoritmi Deterministici**: punti scelti sulla base di info ottenute durante le iterazioni dell'algoritmo
- **Algoritmi Probabilistici**: campionamento avviene in maniera aleatoria (sulla base delle info estratte dall'algoritmo)

Algoritmi di Ottimizzazione Globale

- Algoritmi che usano informazioni globali (convessità, concavità, costante di Lipschitz della funzione obiettivo e dei vincoli, numero di minimi globali, . . .)
- Algoritmi che usano informazioni locali (valori della funzione obiettivo e dei vincoli nei vari punti , valori delle derivate,. . .)

Proprietà di convergenza ottenibili soltanto attraverso un campionamento opportuno

- **Algoritmi Deterministici**: punti scelti sulla base di info ottenute durante le iterazioni dell'algoritmo
- **Algoritmi Probabilistici**: campionamento avviene in maniera aleatoria (sulla base delle info estratte dall'algoritmo)

Algoritmi di Ottimizzazione Globale

- Algoritmi che usano informazioni globali (convessità, concavità, costante di Lipschitz della funzione obiettivo e dei vincoli, numero di minimi globali, . . .)
- Algoritmi che usano informazioni locali (valori della funzione obiettivo e dei vincoli nei vari punti , valori delle derivate,. . .)

Proprietà di convergenza ottenibili soltanto attraverso un campionamento opportuno

- **Algoritmi Deterministici**: punti scelti sulla base di info ottenute durante le iterazioni dell'algoritmo
- **Algoritmi Probabilistici**: campionamento avviene in maniera aleatoria (sulla base delle info estratte dall'algoritmo)

Convergenza Algoritmi

Assunzione 1

L'insieme ammissibile $S \subset R^n$ è un insieme non vuoto e compatto tale che:

$$S = Cl(Int(S))$$

dove $Cl()$ indica la chiusura di un insieme e $Int()$ l'interno di un insieme.

Definizione 1

Un sottoinsieme S di uno spazio topologico F si dice denso (in F) se $cl(S) = F$.

Nota: In uno spazio metrico F, per ogni punto $x \in F$ e per ogni $\epsilon > 0$ esiste un punto $y \in B(x, \epsilon)$ tale che $y \in S$.

Convergenza Algoritmi Deterministici

Teorema 1

Siano

- S insieme che soddisfa l'Assunzione 1;
- C insieme delle funzioni continue su S;
- A_d algoritmo deterministico che usa informazioni locali.

Per ogni funzione $f \in C$ (con $x^\star$ minimo globale di f su S), A_d genera una sequenza $\{x_k\}$ che converge ad $x^\star$

se e solo se

I punti prodotti da A_d (quando $k \to \infty$) generano un insieme denso in S.

Convergenza Algoritmi Deterministici

Teorema 1

Siano

- S insieme che soddisfa l'Assunzione 1;
- C insieme delle funzioni continue su S;
- A_d algoritmo deterministico che usa informazioni locali.

Per ogni funzione $f \in C$ (con $x^\star$ minimo globale di f su S), A_d genera una sequenza $\{x_k\}$ che converge ad $x^\star$

se e solo se

I punti prodotti da A_d (quando $k \to \infty$) generano un insieme denso in S.

Convergenza Algoritmi Probabilistici

Teorema 2

Siano

- S insieme che soddisfa l'Assunzione 1;
- C insieme delle funzioni continue su S;
- A_p algoritmo probabilistico che usa informazioni locali.

Per ogni funzione $f \in C$ (con $x^\star$ minimo globale di f su S), A_p genera una sequenza $\{x_k\}$ che converge ad $x^\star$ con probabilità maggiore di $p \in (0, 1)$

se e solo se

La probabilità che un qualsiasi punto di S appartenga alla chiusura dei punti prodotti da A_p (quando $k \to \infty$) è maggiore di p.

Convergenza Algoritmi

- Convergenza è un requisito minimo e non garantisce efficienza!
- Esempio: Grid Search è algoritmo deterministico convergente, ma lento per $n > 2$
- Per generare griglia $q \times q$ necessari $(q+1)^n$ calcoli di funzione

Problema

Sia f una funzione Lipschitziana in F, ovvero esiste $L > 0$ tale che:

$$|f(x) - f(y)| \leq L\|x - y\| \quad \forall\, x, y \in F.$$

Trovare un punto $\hat{x} \in [0,1]^n$ tale che $f(\hat{x}) - f(x^\star) < \epsilon$ dove

$$f^\star = \min_{x \in [0,1]^n} f(x)$$

Convergenza Algoritmi

- Convergenza è un requisito minimo e non garantisce efficienza!
- Esempio: Grid Search è algoritmo deterministico convergente, ma lento per $n > 2$
- Per generare griglia $q \times q$ necessari $(q+1)^n$ calcoli di funzione

Problema

Sia f una funzione Lipschitziana in F, ovvero esiste $L > 0$ tale che:

$$|f(x) - f(y)| \leq L\|x - y\| \quad \forall\, x, y \in F.$$

Trovare un punto $\hat{x} \in [0,1]^n$ tale che $f(\hat{x}) - f(x^\star) < \epsilon$ dove

$$f^\star = \min_{x \in [0,1]^n} f(x)$$

Convergenza Algoritmi II

Grid Search

- $f(\hat{x}) - f^\star \leq L\sqrt{n}/(2q)$.
  Prova: $\bar{x}$ punto della griglia vicino a $x^\star$. Abbiamo $\|\bar{x} - x^\star\| \leq \sqrt{n}/(2q)$. Dunque otteniamo $f(\hat{x}) - f(x^\star) \leq f(\bar{x}) - f(x^\star) \leq L\sqrt{n}/(2q)$.
- La Grid Search ha bisogno di $(\lceil L\sqrt{n}/(2\epsilon)\rceil + 1)^n$ calcoli di funzioni per ottenere $\hat{x}$ tale che $f(\hat{x}) - f(x^\star) \leq \epsilon$.
- Con $L = 3$, $n = 16$, $epsilon = 0.1$ necessari $36 \cdot 10^{27}$ calcoli di funzione circa
- Se CPU effettua 64 miliardi di calcoli al secondo, necessari 18 miliardi di anni circa

Esplorazione della regione ammissibile effettuata dando priorità alle regioni promettenti

Necessario bilanciare sforzo computazionale tra:

- esplorazione della regione ammissibile
- approssimazione dell'ottimo globale

Convergenza Algoritmi II

Grid Search

- $f(\hat{x}) - f^\star \leq L\sqrt{n}/(2q)$.
  Prova: $\bar{x}$ punto della griglia vicino a $x^\star$. Abbiamo $\|\bar{x} - x^\star\| \leq \sqrt{n}/(2q)$. Dunque otteniamo $f(\hat{x}) - f(x^\star) \leq f(\bar{x}) - f(x^\star) \leq L\sqrt{n}/(2q)$.
- La Grid Search ha bisogno di $(\lceil L\sqrt{n}/(2\epsilon)\rceil + 1)^n$ calcoli di funzioni per ottenere $\hat{x}$ tale che $f(\hat{x}) - f(x^\star) \leq \epsilon$.
- Con $L = 3$, $n = 16$, $epsilon = 0.1$ necessari $36 \cdot 10^{27}$ calcoli di funzione circa
- Se CPU effettua 64 miliardi di calcoli al secondo, necessari 18 miliardi di anni circa

Esplorazione della regione ammissibile effettuata dando priorità alle regioni promettenti

Necessario bilanciare sforzo computazionale tra:

- esplorazione della regione ammissibile
- approssimazione dell'ottimo globale

Convergenza Algoritmi II

Grid Search

- $f(\hat{x}) - f^\star \leq L\sqrt{n}/(2q)$.
  Prova: $\bar{x}$ punto della griglia vicino a $x^\star$. Abbiamo $\|\bar{x} - x^\star\| \leq \sqrt{n}/(2q)$. Dunque otteniamo $f(\hat{x}) - f(x^\star) \leq f(\bar{x}) - f(x^\star) \leq L\sqrt{n}/(2q)$.
- La Grid Search ha bisogno di $(\lceil L\sqrt{n}/(2\epsilon)\rceil + 1)^n$ calcoli di funzioni per ottenere $\hat{x}$ tale che $f(\hat{x}) - f(x^\star) \leq \epsilon$.
- Con $L = 3$, $n = 16$, *epsilon* $= 0.1$ necessari $36 \cdot 10^{27}$ calcoli di funzione circa
- Se CPU effettua 64 miliardi di calcoli al secondo, necessari 18 miliardi di anni circa

Esplorazione della regione ammissibile effettuata dando priorità alle regioni promettenti

Necessario bilanciare sforzo computazionale tra:

- esplorazione della regione ammissibile
- approssimazione dell'ottimo globale

Classi di Algoritmi

- **Incompleti:** usano euristiche efficienti per la ricerca, ma possono terminare in un minimo locale
- **Asintoticamente completi:** garantiscono raggiungimento di un minimo globale (almeno con probabilità 1) se eseguiti per un tempo infinitamente lungo, ma nessun modo per sapere quando un minimo globale è raggiunto
- **Completi:** garantiscono raggiungimento di un minimo globale se eseguiti per un tempo infinitamente lungo, e l'individuazione di un'approssimazione dell'ottimo globale (data una certa tolleranza) dopo un tempo finito
- **Rigorosi:** garantiscono l'individuazione di un'approssimazione dell'ottimo globale (data una certa tolleranza) anche in presenza di errori

Algoritmo Random Search

Schema

P0: Genera randomicamente un punto x_0 su S, poni $x_0^\star = x_0$ e $k = 1$;

P1: Genera randomicamente un punto x_k su S;

P2: Se $f(x_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = x_k$
altrimenti poni $x_k^\star = x_{k-1}^\star$;

P3: Poni $k = k + 1$ e vai a **P1**.

Algoritmo Multistart

Schema

P0: Genera randomicamente un punto x_0 su S, poni $x_0^\star = x_0$ e $k = 1$;

P1: Genera randomicamente un punto x_k su S;

P2: A partire da x_k, esegui un algoritmo di ottimizzazione locale ottenendo y_k;

P3: Se $f(y_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = y_k$;
altrimenti poni $x_k^\star = x_{k-1}^\star$;

P4: Poni $k = k + 1$ e vai a **P1**.

Algoritmo Multistart: Commenti

- Algoritmo Multistart è stato il primo ad essere proposto in letteratura
- Proprietà di convergenza garantite
- Non sfrutta l'informazione generata durante le ricerche locali precedenti
- Numero di minimizzazioni locali può essere molto grande (In generale inefficiente)
- IDEA: Scegliere opportunamente punto iniziale della fase locale

Iterated Local Search/Monotonic Basin Hopping

Caratterizzato da:

- LocalSearch(x) : Algoritmo di ottimizzazione Locale
- Perturb(x): Procedura per perturbare la soluzione x

Algoritmo Iterated Local Search II

Schema

P0: Genera randomicamente un punto x_0 su S;

P1: Poni $x_0^\star = \mathit{LocalSearch}(x_0)$ e $k = 1$;

P2: Poni $x_k = \mathit{Perturb}(x_{k-1}^\star)$;

P3: Poni $y_k = \mathit{LocalSearch}(x_k)$;

P4: Se $f(y_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = y_k$;
altrimenti poni $x_k^\star = x_{k-1}^\star$;

P5: Poni $k = k + 1$ e vai a **P2**.

Iterated Local Search/Monotonic Basin Hopping: Commenti

- Scelta della procedura **Perturb** fondamentale per l'efficienza dell'algoritmo
- Se punto generato in intorno **troppo piccolo** $\Rightarrow$ bloccato in un minimo locale
- Se punto generato in intorno **troppo grande** $\Rightarrow$ equivalente al Multistart
- **Approccio classico:** scegliere $\delta > 0$ e generare randomicamente il punto $x_k \in B(x^\star_{k-1}, \delta)$
- **LocalSearch** e **Perturb** dipendono dal Problema

Algoritmo Simulated Annealing

- Definire algoritmi probabilistici che generano punti secondo funzioni densità di probabilità non costanti su S
- IDEA: Utilizzare funzioni densità di probabilità concetrate intorno ai minimi globali
- **Annealing:** materiale (metallo) viene prima riscaldato poi lentamente raffreddato. Obiettivo è raggiungere stato con minima energia (garantisce particolari proprietà fisiche)
- **Fisica:** più alta è la temperatura, più facile è raggiungere stati non stabili
- **Ottimizzazione:** Riproduciamo il fenomeno fisico
  - Valore temperatura corrente in T
  - Riduci T lentamente
  - Usa T per influenzare la probabilità ottenere soluzioni peggiori

Algoritmo Simulated Annealing

- s stato del sistema e $E(s)$ energia associata
- All'equilibrio termico: densità di probabilità proporzionale a $e^{-E(s)/\mathcal{K}T}$ ($\mathcal{K}$ costante di Boltzmann)
- Nel nostro caso: stato x, Energia $E(x) = f(x) - f^\star \geq 0$ e $\mathcal{K} = 1$
- Quando $T \to 0$ probabilità stati ad energia positiva tende a zero - solo stati ad energia nulla possibili (densità si concentra attorno ai minimi globali)
- Densità di probabilità:

$$\hat{p}_T(x) = \frac{e^{-[f(x)-\hat{f}]_+/T}}{\int_D e^{-[f(x)-\hat{f}]_+/T} dx}$$

  dove $[f(x) - \hat{f}]_+ = \max\{0, f(x) - \hat{f}\}$
- $\hat{f}$ approssimazione dell'ottimo globale
- Quando $T \to 0$ densità si concentra attorno a punti $x \in S$ tali che $f(x) \leq \hat{f}$

Generazione Punti (Metodo di Von Neumann)

Per generare punti con densità di probabilità $\hat{p}_T(x)$:

- T scalare positivo, $\hat{f} \geq f^\star$ approssimazione dell'ottimo globale
- α scalare scelto randomicamente su [0,1], x vettore scelto randomicamente su S
- Se $\alpha \leq e^{-[f(x)-\hat{f}]_+/T}$ allora punto x realizzazione della variabile aleatoria distribuita secondo $\hat{p}_T(x)$

Algoritmo Simulated Annealing (Base)

Schema

P0: Scegli una temperatura iniziale T_0, genera randomicamente un punto x_0 su S, poni $x_0^\star = x_0$ e $k = 1$;

P1: Genera randomicamente un punto $x_k \in B(x_{k-1}, \delta)$ ed uno scalare $\alpha_k \in [0, 1]$;

P2: Se $\alpha_k > e^{-[f(x_k) - f(x_{k-1}^\star)]_+ / T_{k-1}}$ allora poni $x_k = x_{k-1}$ e vai a **P5**;

P3: Se $f(x_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = x_k$;
altrimenti poni $x_k^\star = x_{k-1}^\star$;

P4: Scegli $T_k \in (0, T_{k-1}]$;

P5: Poni $k = k + 1$ e vai a **P1**.

Algoritmo Simulated Annealing (Con Ricerca Locale)

Schema

P0: Scegli una temperatura iniziale T_0, genera randomicamente un punto x_0 su S, poni $x_0^\star = x_0$ e $k = 1$;

P1: Genera randomicamente un punto x_k su S ed uno scalare $\alpha_k \in [0,1]$;

P2: Se $\alpha_k > e^{-[f(x_k)-f(x_{k-1}^\star)]_+/T_{k-1}}$ allora poni $x_k^\star = x_{k-1}^\star$ e vai a **P6**;

P3: A partire da x_k, esegui un algoritmo di ottimizzazione locale ottenendo y_k;

P4: Se $f(y_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = y_k$;
altrimenti poni $x_k^\star = x_{k-1}^\star$;

P5: Scegli $T_k \in (0, T_{k-1}]$;

P6: Poni $k = k+1$ e vai a **P1**.

Algoritmi Genetici

- Analogie con processo evolutivo (selezione naturale, mutazione, crossover)
- Ad ogni iterazione una “popolazione” di punti viene generata
- Selezione, mutazione e crossover utilizzati per ottenere nuovi punti
- **Selezione:** soluzioni migliori mantenute
- **Mutazione:** modifica casuale di alcune soluzioni
- **Crossover:** soluzioni combinate tra loro

- generazione randomica di una popolazione iniziale
- creazione di una nuova popolazione ad ogni iterazione (individui della popolazione corrente usati per creare la generazione successiva)

Algoritmi Genetici

- Analogie con processo evolutivo (selezione naturale, mutazione, crossover)
- Ad ogni iterazione una "popolazione" di punti viene generata
- Selezione, mutazione e crossover utilizzati per ottenere nuovi punti
- **Selezione:** soluzioni migliori mantenute
- **Mutazione:** modifica casuale di alcune soluzioni
- **Crossover:** soluzioni combinate tra loro

- generazione randomica di una popolazione iniziale
- creazione di una nuova popolazione ad ogni iterazione (individui della popolazione corrente usati per creare la generazione successiva)

Algoritmi Genetici: Generazione Nuova Popolazione

- membri valutati secondo un valore di fitness;
- ordinamento di tali individui sulla base dei valori di fitness;
- individui promettenti selezionati come genitori;
- nuovi individui generati effettuando cambiamenti casuali su un singolo genitore (mutazione) oppure combinando opportunamente le caratteristiche di una coppia di genitori (crossover);
- nuovi individui e genitori formano la nuova popolazione.

Algoritmi Genetici

Schema

P0: Genera Randomicamente una popolazione iniziale S_0, e poni $k = 1$

P1: Seleziona individui migliori (alto valore di fitness) per nuova generazione

P2: Produci nuovi individui attraverso mutazione e crossover

P3: Sostituisci individui peggiori (basso valore di fitness) con nuovi individui e ottieni S_k

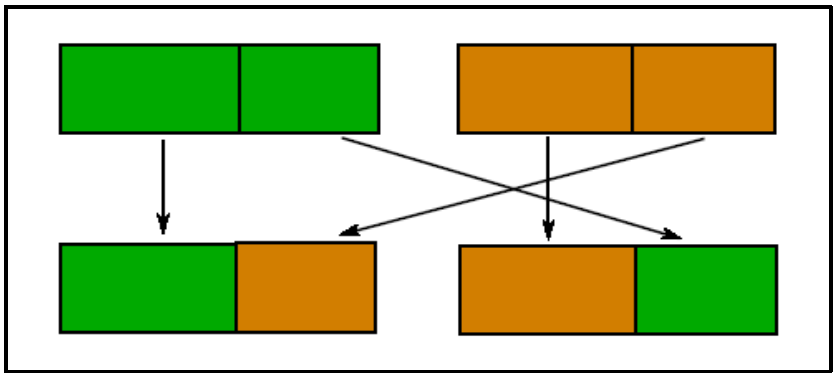
P4: Poni $k = k + 1$ e vai a **P1**.

Selezione dei Genitori

- **Elitist strategy:** Seleziona i migliori m individui
- **Tournament:** Seleziona il migliore tra k individui selezionati randomicamente (ripeti m volte)
- **Roulette wheel:** Seleziona m individui randomicamente (favorendo quelli con alto valore di fitness)

Crossover

- **Single-Point Crossover:** Prendi due individui e taglia le soluzioni in un punto a caso. Combina le soluzioni.
- **Two-Point Crossover :** Prendi due individui e taglia le soluzioni in due punti a caso. Combina le soluzioni.
- **Altri metodi:** Cut and Splice, Uniform Crossover, . . .



Mutazione

- **Obiettivo:** Perturbare soluzione per garantire diversificazione
- **Troppa diversificazione:** comportamento randomico
- **Poca diversificazione:** bloccato in minimo locale

Tabu Search

- **IDEA:** Accettare soluzioni peggiorative
- **PROBLEMA:** Scelta di soluzione peggiorativa può portare a Loop
- **SOLUZIONE:** Accetta soluzioni peggiorative $+$ Evita passaggio per soluzioni già esaminate
- Memorizzazione di info relative al processo di ricerca per garantire uscita da Loop
- Ricerca effettuata in un intorno $N'(x)$ da cui rimosse soluzioni tabu
- **Tabu List:** Memoria del processo di ricerca

Tabu List

- Scelta lunghezza TL fondamentale per corretto funzionamento dell'algoritmo
- **Regole statiche:** Lunghezza invariata per tutto l'algoritmo (E.g. valore tra 5 e 20 oppure $\sqrt{n}$)
- **Regole dinamiche:** Lunghezza varia durante l'esecuzione (E. g. fissati due valori TLmin TLmax, ogni soluzione ha periodo di permanenza $TLmin \leq l_{TL} \leq TLmax$)

Tabu Search

Schema

P0: Genera randomicamente un punto x_0 su S, poni $x_0^\star = x_0$ e $k = 1$;
P1: Poni x_k uguale al miglior punto trovato in $N'(x_{k-1})$;
P1: Aggiorna Tabu List;
P3: Se $f(x_k) < f(x_{k-1}^\star)$ allora poni $x_k^\star = x_k$;
altrimenti poni $x_k^\star = x_{k-1}^\star$;
P5: Poni $k = k + 1$ e vai a **P1**.

Rilassamenti

Consideriamo il problema (P) (non-convesso):

$$\begin{aligned} &\min f(x) \\ &x \in X \end{aligned}$$

Assumiamo che il minimo esista e sia finito.

Possiamo usare un rilassamento (convesso) (R):

$$\begin{aligned} &\min g(x) \\ &x \in Y \end{aligned}$$

(R) rilassamento di (P) iff:

- $X \subseteq Y$
- $g(x) \leq f(x) \; \forall \, x \in X$

Inviluppi Convessi

Definition 1

Dato un insieme $S \subseteq R^n$ una **sottostima convessa** di una funzione $f : S \to R$ é una funzione g tale che:

(a) $g(x)$ convessa;

(b) $g(x) \leq f(x) \quad \forall\ x \in S$.

Definition 2

Dato un insieme $S \subseteq R^n$ un **inviluppo convesso** di una funzione $f : S \to R$ é una funzione g tale che:

(a) $g(x)$ sottostima convessa di f;

(b) $g(x) \geq h(x) \quad \forall\ x \in S \quad$ and $\quad \forall\ h$ sottostima convessa di f.

Inviluppi Convessi

Definition 1

Dato un insieme $S \subseteq R^n$ una **sottostima convessa** di una funzione $f : S \to R$ é una funzione g tale che:

(a) $g(x)$ convessa;

(b) $g(x) \leq f(x) \quad \forall\ x \in S$.

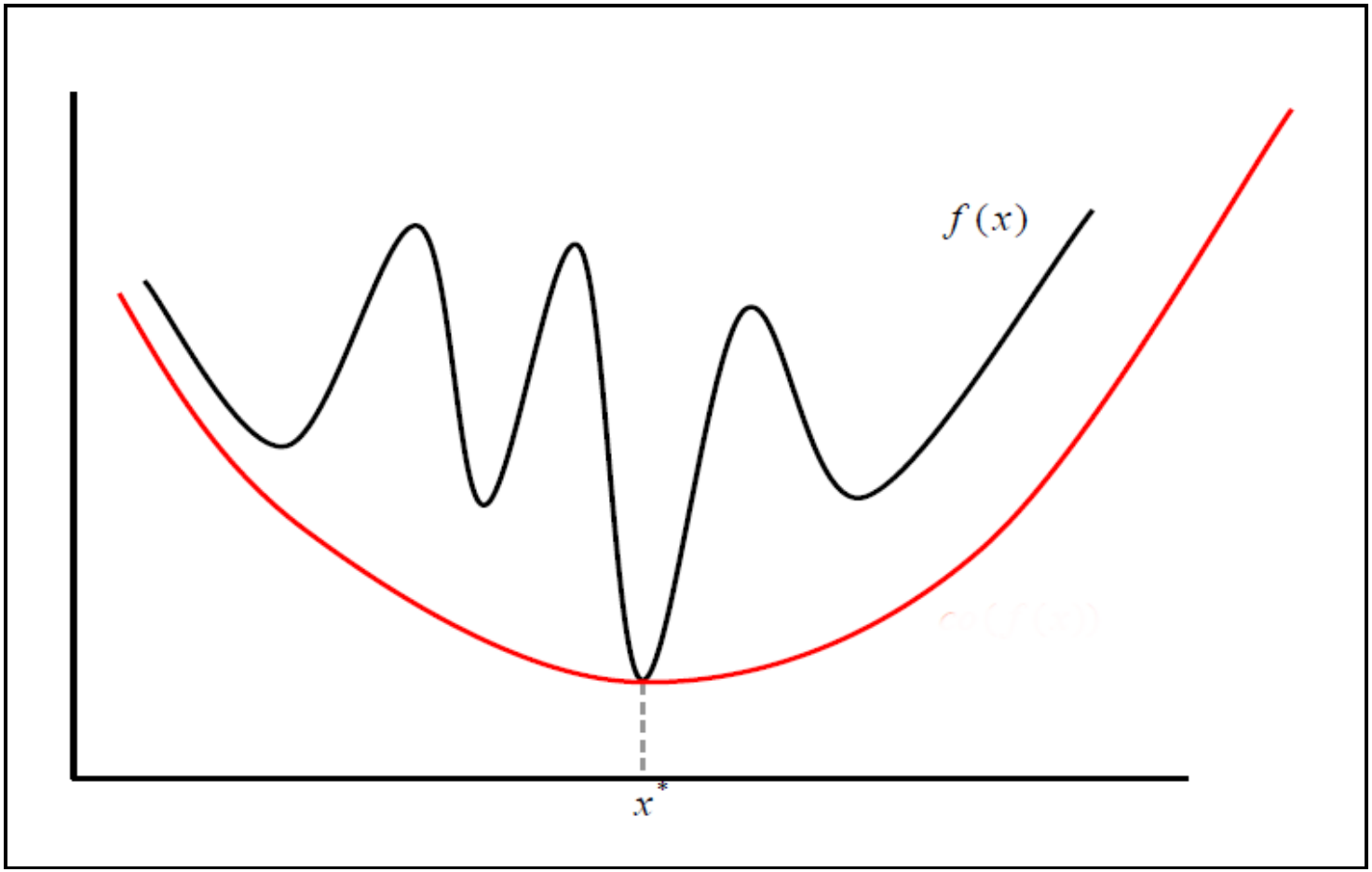
Definition 2

Dato un insieme $S \subseteq R^n$ un **inviluppo convesso** di una funzione $f : S \to R$ é una funzione g tale che:

(a) $g(x)$ sottostima convessa di f;

(b) $g(x) \geq h(x) \quad \forall\ x \in S \quad$ and $\quad \forall\ h$ sottostima convessa di f.

Inviluppi Convessi II



Inviluppi Convessi III

Definition 3

Dato un insieme $S \subseteq R^n$ l'**epigrafo** $epi(f)$ di una funzione $f : S \to R$ é:

$$epi(f) = \{(x, r) \in S \times R : \ r \geq f(x)\} \ . \tag{5}$$

Definition 4

Dato un insieme $S \subseteq R^n$ l' *inviluppo convesso* g di $f : S \to R$ viene espresso come segue:

$$g(x) = \inf\{ \ r : (x, r) \in conv(epi(f)) \ \} \ . \tag{6}$$

Inviluppi Convessi III

Theorem 5

Se $f : S \to R$ continua su $S \subset R^n$ insieme convesso e compatto, allora l'inviluppo convesso g é tale che

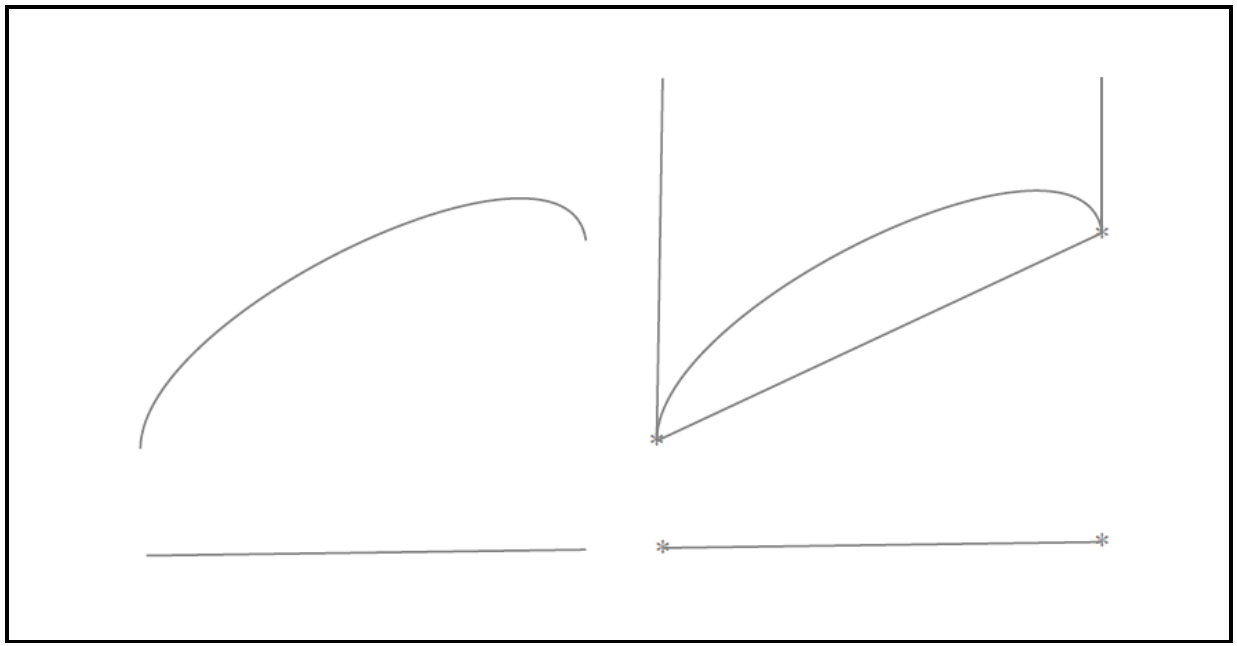
(i) $\min_{x \in S} g(x) = \min_{x \in S} f(x)$;

(ii) $\arg\min_{x \in S} g(x) \supseteq \arg\min_{x \in S} f(x)$;

(iii) $g(x) = f(x)$ *nei punti estremi di S.*

Inviluppo Convesso

- **Esempio:** Sia $f : S \to R$ una funzione concava su un intervallo chiuso $S = [l, u] \subset R$ .Allora l'inviluppo convesso g di f é il segmento congiungente $(l, f(l))$ e $(u, f(u))$.



Intro

Consideriamo il problema (P):

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in X \end{aligned} \tag{7}$$

con $X \subseteq R^n$, e assumiamo che il minimo di f su X esiste ed é finito.

Definition 6

Sia $X \subseteq R^n$ e I sia un insieme finito di indici. Un insieme $\{X_i \mid i \in I\}$ di sottoinsiemi di X si dice *partizione* di X se

$$X = \cup_{i \in I} X_i;$$

$$X_i \cap X_j = \partial X_i \cap \partial X_j \quad \forall\, i,j \in I,\ i \neq j$$

Intro

Consideriamo il problema (P):

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & x \in X \end{aligned} \tag{7}$$

con $X \subseteq R^n$, e assumiamo che il minimo di f su X esiste ed é finito.

Definition 6

Sia $X \subseteq R^n$ e I sia un insieme finito di indici. Un insieme $\{X_i \mid i \in I\}$ di sottoinsiemi di X si dice *partizione* di X se

$$X = \cup_{i \in I} X_i;$$
$$X_i \cap X_j = \partial X_i \cap \partial X_j \quad \forall\, i,j \in I,\ i \neq j$$

Branch and Bound per l'Ottimizzazione Globale

Branch and Bound:

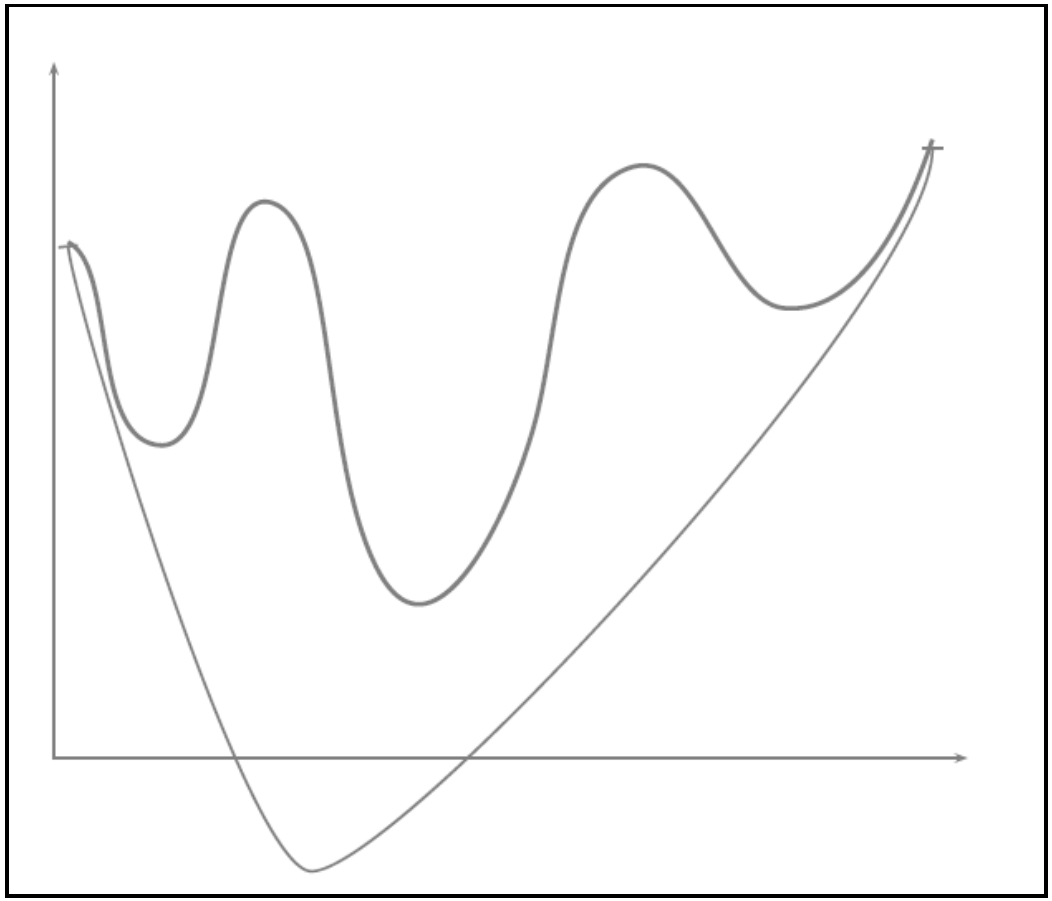
Inizilizzazione: $M_0 = X$, $PX_0 = \{M_0\}$, $L_0 = L(M_0)$ and $U_0 = U(M_0)$
Se $U_0 = L_0$ allora STOP. U_0 ottimo per P;

Iterazione k

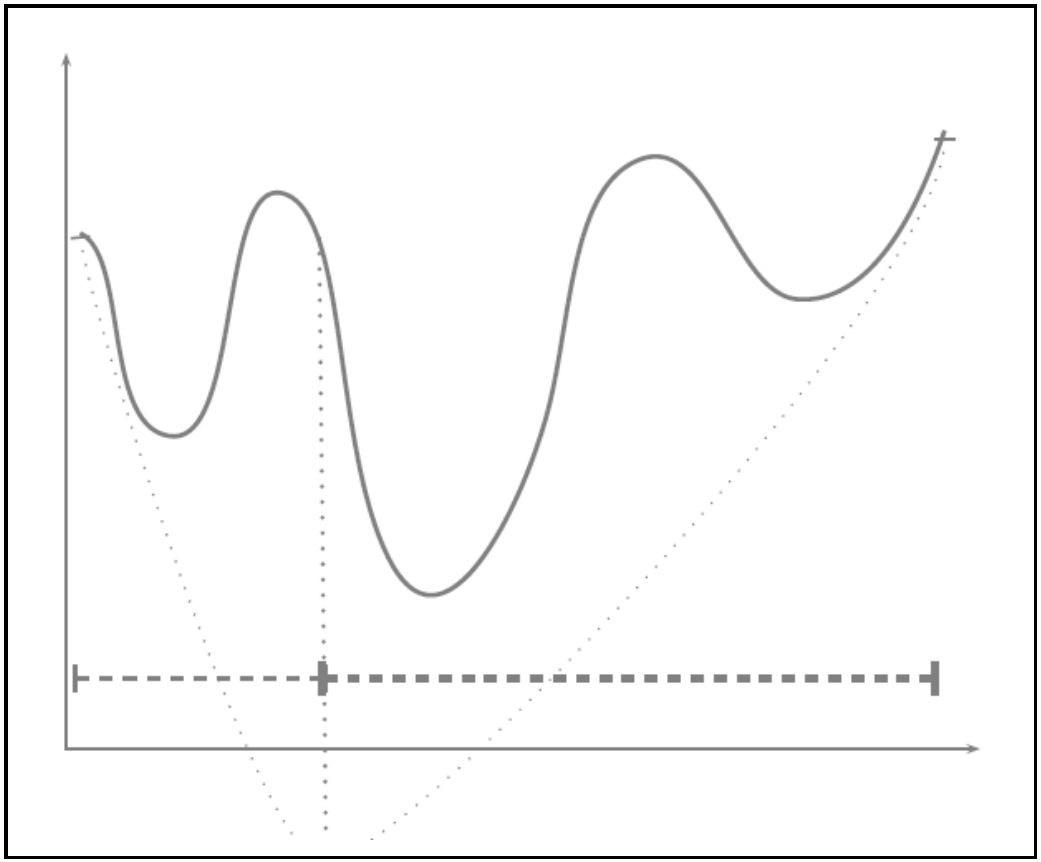
1. *Partitioning Step*: Costruisco una partizione PM_{k-1} del sottoinsieme M_{k-1} e la aggiungo alla lista degli elementi partizionati $PX_k = (PX_{k-1} \setminus M_{k-1}) \cup PM_{k-1}$ da esplorare;
2. *Bounding Step*: Per ogni regione $M \in PM_{k-1}$ determino lower and upper bounds $L(M)$ e $U(M)$, tali che
$$L(M) \leq f(x) \leq U(M), \ \forall\, x \in M;$$
3. *Global Bounding Step*: Assegno L_k e U_k come segue:
$$L_k = \min\{L(M) \mid M \in PX_k\}, \quad U_k = \min\{U(M) \mid M \in PX_k\};$$
4. *Fathoming Step*: Rimuovo ogni $M \in PX_k$ da PX_k per cui:
$$L(M) \geq U_k;$$
5. *Termination e Selection Step*:
Se $U_k = L_k$ allora STOP. U_k ottimo per P;

   Altrimenti, seleziono una regione M_k dalla lista PX_k, Set $k = k + 1$ e vai allo **STEP 1**.

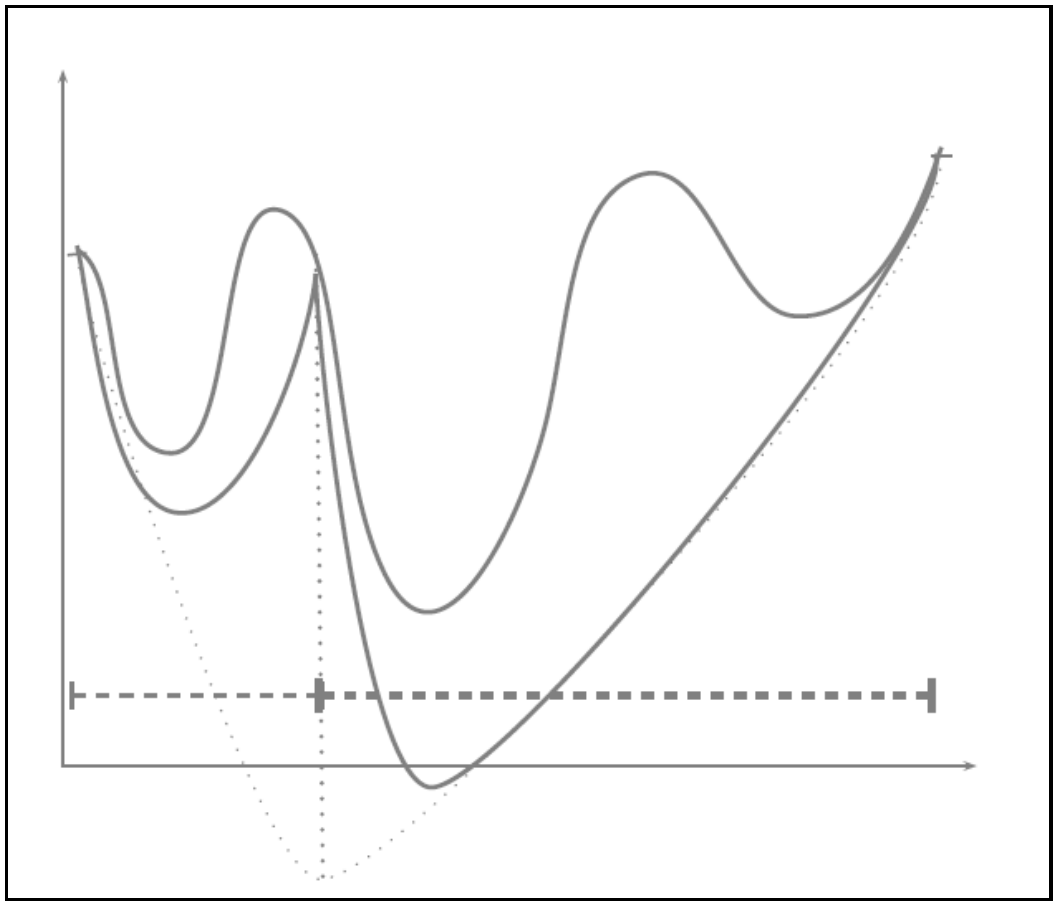
Branch and Bound I



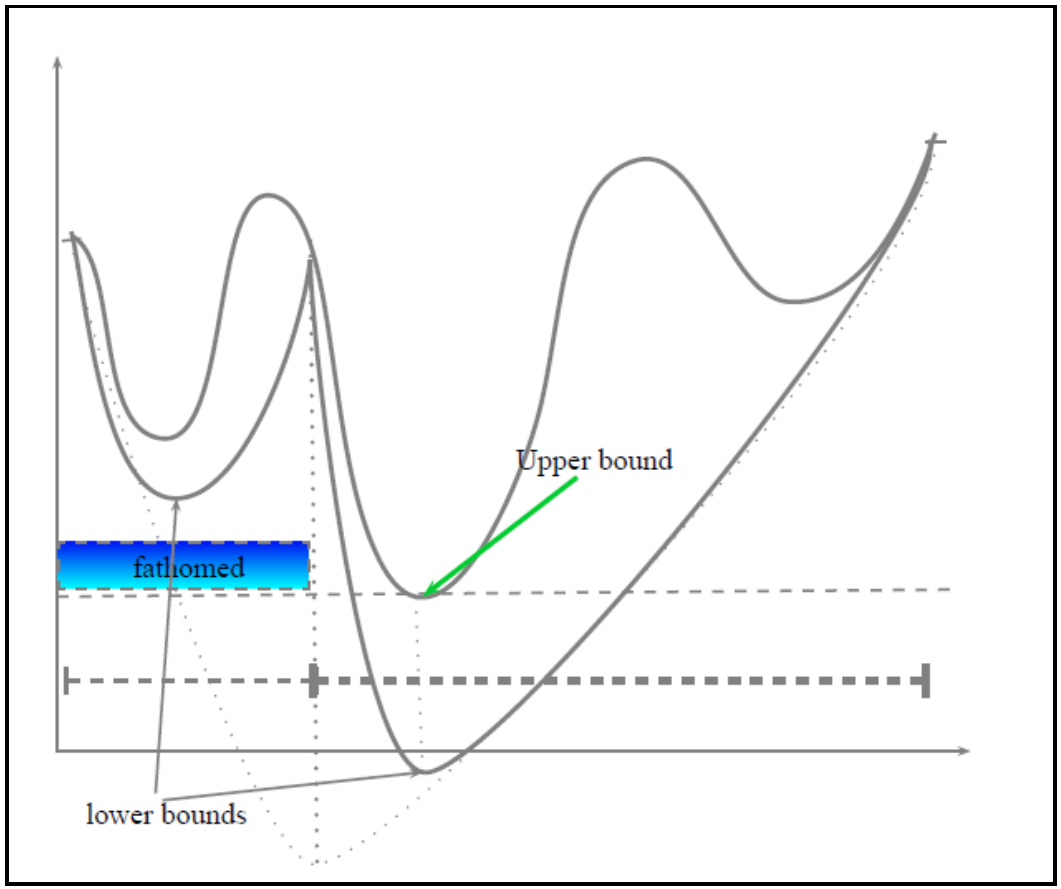
Branch and Bound II



Branch and Bound II



Branch and Bound



Branch and Bound: Convergenza

Definition 7

Un algoritmo di Branch-and-Bound é *finito* se $L_k = U_k$ per $k < \infty$.

Quando non c'é terminazione finita, necessario studiare comportamento al limite.

Definition 8

Un algoritmo di Branch-and-Bound é *convergente* se

$$\lim_{k \to \infty} |U_k - L_k| = 0.$$

Convergenza dipende da tre scelte:

1. Partitioning;
2. Bounding;
3. Selection.

Branch and Bound: Convergenza

Definition 7

Un algoritmo di Branch-and-Bound é *finito* se $L_k = U_k$ per $k < \infty$.

Quando non c'é terminazione finita, necessario studiare comportamento al limite.

Definition 8

Un algoritmo di Branch-and-Bound é *convergente* se

$$\lim_{k \to \infty} |U_k - L_k| = 0.$$

Convergenza dipende da tre scelte:

1. Partitioning;
2. Bounding;
3. Selection.