UNIVERSITÀ DEGLI STUDI DI CAGLIARI
DIPARTIMENTO DI MATEMATICA E INFORMATICA
DOTTORATO DI RICERCA IN MATEMATICA - XIX CICLO

Un nuovo metodo per l'ottimizzazione globale

Settore scientifico disciplinare:
ANALISI NUMERICA - MAT/08

Tutore:
Prof. Marco Gaviano

Tesi di Dottorato di:
Anna Maristella Steri

Anno Accademico 2005-2006

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La risoluzione di problemi decisionali esige, in ambito scientifico, una quantificazione che indirizzi il processo decisionale alla scelta ottimale. Numerosi problemi decisionali che si incontrano in ambito fisico, chimico, economico, ingegneristico e via dicendo, sono spesso modellizzati come problemi di ottimizzazione di una funzione reale, che può rappresentare ad esempio la prestazione o il costo di un sistema in particolari condizioni; questa funzione, detta *funzione obiettivo*, generalmente è di tipo non lineare ed è spesso soggetta a dei vincoli, che garantiscono l'accettabilità della soluzione individuata.
L'ottimizzazione globale, in particolare, ha come obiettivi:

- l'analisi di modelli decisionali non lineari aventi molteplici soluzioni ottimali;
- la progettazione e lo studio di algoritmi di risoluzione efficienti in grado di individuare la soluzione (spesso non unica) globalmente migliore.

Dal punto di vista matematico, il problema generale dell'ottimizzazione globale (**GOP**) consiste nell'individuare il minimo globale di una funzione $f : S \to \mathbb{R}$, con $S \subseteq \mathbb{R}^n$ chiamato solitamente *insieme ammissibile* o *spazio di ricerca*; il problema **GOP** si indica nel modo seguente:

$$\underset{x \in S}{\text{minimizzare}} \quad f(x).$$

Con la stessa notazione si indica anche il problema della minimizzazione locale della f (**LOP**).

Nel caso in cui la funzione obiettivo abbia un unico punto di minimo locale, che coincide con la soluzione di **GOP**, allora f è detta *mono-estremale*. Questa proprietà deriva dalla particolare struttura del modello matematico considerato, ad esempio la stretta convessità di f e la convessità di S, e si verifica tipicamente quando ci si trova "sufficientemente vicino" alla regione di attrazione di un minimo. Questa proprietà non rappresenta però una caratteristica generale dei problemi di ottimizzazione: molto frequentemente il numero dei minimi locali non è noto e può essere anche molto grande; inoltre le soluzioni locali possono essere molto distanti da quella globale. In questo caso risolvere il problema **GOP** significa individuare la migliore tra tutte le soluzioni, evitando di restare "intrappolati" in un minimo locale non globale.

Le classi di problemi da risolvere nell'ambito dell'ottimizzazione, sia locale che globale, sono svariate e molto diverse tra loro: dai problemi di ottimizzazione combinatoria alla programmazione quadratica, dalla minimizzazione concava alla programmazione differenziale convessa, dai problemi di mini-max all'ottimizzazione lipschitziana. Di conseguenza, gli approcci proposti per la risoluzione di questi problemi sono molto vari: se da un lato una strategia molto generale potrebbe essere adatta in tutti i casi, risultando ovviamente inefficiente nei casi specifici, d'altra parte metodi strettamente specializzati sono applicabili solo per la classe di problemi per cui sono stati progettati.

Algoritmi efficienti per la risoluzione del problema **LOP** sono stati proposti già da tempo: il *metodo di più ripida discesa* ad esempio risale addirittura a Cauchy (1800). Questi metodi non sono però in grado di individuare l'ottimo globale di f, per cui è necessario introdurre delle strategie globali opportune, che forniscano garanzie di convergenza alla soluzione e magari sfruttino l'efficienza delle strategie locali. I primi algoritmi per la risoluzione del problema **GOP** risalgono alla prima metà del Novecento. Negli anni Settanta sono state proposte le prime strategie di tipo euristico tentando, anche se con scarso successo, di estendere algoritmi convergenti nel caso unidimensionale a dimensioni superiori. Le prime raccolte di lavori sull'ottimizzazione globale sono dovute a Dixon e Szegö (1975, 1978): da allora sono decine i libri e centinaia gli articoli sull'argomento pubblicati su numerose riviste scientifiche.

Si osservi che i metodi di ottimizzazione proposti, in quanto procedure numeriche, non sono generalmente in grado di individuare la soluzione esatta del problema, ma solo un'approssimazione numerica dei punti di minimo globale, nonché del minimo globale stesso.

A seconda che facciano uso o meno di elementi probabilistici, i metodi sviluppati per la risoluzione di **GOP** possono essere suddivisi in linea di massima in due grandi classi:

1. metodi deterministici;

2. metodi stocastici.

Tipicamente, alla classe dei metodi deterministici appartengono tutti i metodi del tipo *Branch and Bound*, come ad esempio il *metodo Grid Search*, i metodi di partizione su intervalli, i metodi basati sulle curve di Peano e le strategie di partizione per funzioni lipschitziane; alla stessa classe appartengono anche alcuni metodi che utilizzano perturbazioni della funzione obiettivo, come il *metodo di tunneling* o il *metodo filled function*. Questi metodi forniscono un'assoluta garanzia di successo ma richiedono ipotesi restrittive sulla funzione.

I metodi stocastici richiedono in genere ipotesi più deboli rispetto ai primi e assicurano una convergenza probabilistica all'ottimo globale. A questa classe appartengono ad esempio i *metodi Two-Phase*, i metodi concettuali di tipo *Random Search*, i metodi di tipo *Simulated Annealing* e i metodi *Random Directions*. Da un lato i metodi Two-Phase, come il Multistart, e i metodi Random Directions

si basano su ricerche locali e mirano all'individuazione di tutti i minimi locali della funzione. Se il numero dei minimi è molto elevato, è preferibile cercare di determinare direttamente il minimo globale; questa è l'idea alla base dei metodi concettuali Random Search e del Simulated Annealing.

Esistono in letteratura metodi basati su procedure stocastiche che, in particolari condizioni, risultano competitivi dal punto di vista empirico ma che non forniscono garanzie di convergenza. Questi metodi, detti *euristici*, comprendono ad esempio tutti quegli approcci che simulano i processi dell'evoluzione biologica di un sistema, come la selezione naturale, e si applicano a popolazioni di punti che vengono ricombinati sequenzialmente fino a generare la soluzione ottima. Tra questi citiamo i *metodi evolutivi* e gli *algoritmi genetici*.

Le principali difficoltà di risoluzione del problema **GOP** sono legate al fatto che spesso non è nota la struttura della funzione da minimizzare, quindi non si conosce il numero e la distribuzione dei minimi, né la misura delle loro regioni di attrazione. Inoltre, al crescere della dimensione, generalmente aumenta anche il numero dei minimi locali, con conseguente riduzione della regione di attrazione di ogni minimo e quindi della probabilità di individuare l'ottimo globale. Questo significa che la difficoltà di risoluzione del problema può essere anche molto elevata; in termini tecnici il problema in questione è di tipo *NP-hard*, cioè ci si attende che esistano istanze del problema, ossia dati in input, che richiedono uno sforzo computazionale per la risoluzione di **GOP** che cresce esponenzialmente con la dimensione dell'istanza. Alle sopracitate problematiche da superare, nell'applicazione di metodi stocastici basati su ricerche locali si aggiungono le seguenti:

- evitare di ritrovare gli stessi minimi;
- disporre di un efficiente criterio di stop.

La nostra attenzione si concentrerà esclusivamente sulla risoluzione del problema **GOP** nel caso in cui la funzione obiettivo f sia continua e l'insieme ammissibile S sia un compatto di $\mathbb{R}^n$.

Nei capitoli 1 e 2 della presente tesi si esamineranno le principali problematiche relative al problema **GOP**. In particolare, data l'importanza dei minimi locali nel processo di minimizzazione, nel capitolo 1 si darà una descrizione delle condizioni che caratterizzano dal punto di vista analitico i minimi locali di una funzione per problemi vincolati e non, nonché delle condizioni di ottimalità globale in alcune situazioni particolari. Nello stesso capitolo si metterà in evidenza come non esista una caratterizzazione generale del minimo globale simile all'annullarsi del gradiente nel caso locale. Nel capitolo 2 saranno esposte brevemente le caratteristiche generali dei metodi di ottimizzazione locale e globale.

Il capitolo 3 sarà dedicato alla descrizione dei più noti metodi di tipo stocastico, mentre una trattazione generale della complessità computazionale degli algoritmi per la risoluzione del problema **GOP** sarà oggetto del capitolo 4.

Nel capitolo 5 si presenterà un nuovo metodo tramite il quale si è cercato di superare le difficoltà precedentemente elencate. Questo metodo, partendo da un

minimo locale, applica una procedura di ricerca locale sia ad ogni punto a cui corrisponde un valore inferiore all'ottimo corrente, sia a punti con valore superiore; in quest'ultimo caso l'applicazione della ricerca locale avviene però con una probabilità d_i, calcolata in modo da minimizzare il numero di valutazioni di funzione necessario ad individuare un nuovo minimo locale.

In particolare, verrà effettuato uno studio sulle proprietà teoriche "a priori" del metodo proposto in due casi:

a) l'algoritmo può essere eseguito per un numero infinito di iterazioni;

b) l'algoritmo viene eseguito per un numero finito di iterazioni.

In entrambi i casi la funzione che rappresenta il numero medio di valutazioni della f è molto complessa non solo da minimizzare ma anche da studiare. Si ricorre pertanto ad un'analisi locale dell'algoritmo nel passaggio da un minimo ad uno nuovo. Si dimostra che, dal punto di vista teorico, in entrambi i casi analizzati, i valori ottimali della probabilità d_i sono i valori estremi dell'intervallo $[0, 1]$.

Sulla base delle relazioni teoriche ottenute, si presenterà una prima implementazione del metodo proposto. I risultati ottenuti, confrontati con versioni dell'algoritmo in cui non si è effettuata alcuna analisi per la riduzione del numero di valutazioni di funzione, hanno dimostrato che il nuovo approccio è positivo.

Il lavoro portato avanti ha evidenziato vari aspetti da approfondire, in particolare:

- l'individuazione, tramite le relazioni esposte nella tesi, di proprietà generali che "a priori" consentono di stabilire il comportamento ottimale di un algoritmo per una determinata famiglia di problemi;

- la ricerca dei parametri ottimali da assegnare ad un algoritmo per l'implementazione del metodo proposto.

Capitolo 1

Condizioni di ottimalità

Numerosi problemi applicativi che si incontrano in ambito scientifico possono essere modellizzati come problemi di ottimizzazione globale, ossia come problemi in cui si deve determinare il minimo o il massimo globale di una funzione a valori reali $f : \mathbb{R}^n \to \mathbb{R}$. Risulta quindi importante la messa a punto e lo studio di metodi numerici che siano in grado di individuare l'ottimo globale di una simile funzione.

Le difficoltà nella risoluzione di questo tipo di problemi dipendono dalle proprietà della funzione e dell'insieme di definizione. Per quanto riguarda gli algoritmi di risoluzione del problema dell'ottimizzazione globale, la situazione è soddisfacente solo per funzioni relativamente semplici.

In questo capitolo, dopo aver definito il problema dell'ottimizzazione globale, si concentrerà l'attenzione sullo studio delle condizioni di ottimalità locale e globale per problemi vincolati e non, per concludere con alcune recenti caratterizzazioni del minimo globale.

1.1 Definizione del problema

Dal punto di vista matematico, il problema dell'*ottimizzazione globale* in $\mathbb{R}^n$ può essere così formulato:

$$\text{determinare} \quad x^* \in S, \quad \text{tale che} \quad f(x^*) \leq f(x), \quad \forall x \in S,$$

dove $f : S \to \mathbb{R}$ è una funzione definita su un insieme $S \subseteq \mathbb{R}^n$. Tale problema verrà indicato nel modo seguente:

Problema 1.1.1.

$$\begin{aligned} &\text{minimizzare} \quad f(x), \\ &x \in S. \end{aligned} \tag{1.1}$$

La funzione f prende il nome di *funzione obiettivo* e l'insieme S viene detto solitamente *insieme ammissibile*.

Con la notazione (1.1) viene indicata anche la *minimizzazione locale*, che consiste nell'individuare un $x^* \in S$ tale che

$$f(x^*) \leq f(x), \ \forall x \in S : \|x^* - x\| \leq \varepsilon,$$

con $\|\cdot\|$ norma euclidea ed ε un'opportuna costante positiva.

La nostra trattazione riguarderà esclusivamente problemi di minimizzazione. Infatti, il problema

$$\begin{aligned} &\text{massimizzare} \quad f(x) \\ &x \in S, \end{aligned} \tag{1.2}$$

è riconducibile al problema 1.1.1 in quanto:

$$\max f(x) = -\min(-f(x)), \quad x \in S.$$

L'insieme S è spesso definito come

$$S = \{x \in \mathbb{R}^n : g_i(x) \leq 0, \ i = 1, \ldots, p\},$$

con $g_i : A \to \mathbb{R}$ e $A \supseteq S$ (di frequente $A = \mathbb{R}^n$). In tal caso le funzioni g_i prendono il nome di *vincoli*.

Il problema di ottimizzazione 1.1.1 in cui $S \subset \mathbb{R}^n$, è usualmente chiamato *problema di ottimizzazione vincolata*. Se le funzioni f e $g_i, \ i = 1, \ldots, p$, sono continue e S è un insieme connesso, il problema 1.1.1 è detto *problema di programmazione* (o ottimizzazione) *non lineare*. Casi importanti, sia per le applicazioni che per le tecniche e i risultati relativi, sono i seguenti:

- se la funzione f è convessa ed S è un insieme convesso, si ha un *problema di programmazione convessa*;
- se la funzione f è concava su S un insieme convesso, si parla di *problema di programmazione concava*;
- se S è un poliedro, cioè le funzioni g_i sono lineari, il problema è detto *linearmente vincolato*; in questo caso, si tratta di risolvere il problema nella forma:
$$\min\{f(x) \mid Ax \leq b\},$$
dove $A \in \mathbb{R}^{m \times n}$ è una matrice assegnata, $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$;

- se la funzione f è quadratica e i vincoli g_i sono lineari, si ha un *problema di programmazione quadratica*. In tal caso il problema assume la forma seguente:
$$\min\{c^T x + \frac{1}{2} x^T Q x \mid Ax \leq b\},$$
dove $Q \in \mathbb{R}^{n \times n}$ è una matrice simmetrica, $A \in \mathbb{R}^{m \times n}$ una matrice assegnata, $b \in \mathbb{R}^m$ e $c, x \in \mathbb{R}^n$;

- se le funzioni g_i e la funzione obiettivo f sono lineari, si parla di *ottimizzazione lineare*;

- se alcune delle funzioni f e g_i sono discontinue o se alcune delle variabili assumono solo valori discreti si ha un problema di *programmazione discreta*; se in particolare le variabili assumono solo valori interi il problema è detto di *programmazione intera* o di *programmazione booleana* se i valori assunti dalle variabili sono solo 0 o 1.

Quando infine $S = \mathbb{R}^n$, si parla di *problema di ottimizzazione non vincolata*.

1.2 Condizioni di ottimalità locale

La risoluzione numerica di un problema di ottimizzazione globale richiede spesso l'utilizzo di procedure di minimizzazione locale. Vista l'importanza che rivestono nella teoria dell'ottimizzazione, vediamo come si possono caratterizzare dal punto di vista analitico i minimi locali di una funzione (cfr. [34]).
Sia f una funzione continuamente differenziabile in un intorno di un punto $x^* \in S$ e sia $d \in \mathbb{R}^n$ un vettore tale che $d^T \nabla f(x^*) < 0$, dove $d^T \nabla f(x^*)$ è la derivata direzionale di f in x^* nella direzione d. Una direzione d per cui $d^T \nabla f(x^*) < 0$ prende il nome di *direzione di discesa*. Si definisca la funzione $h(x) := f(x^* + \lambda d)$, con $\lambda \in \mathbb{R}^+$ e x^*, d fissati; la funzione $h(x)$ descrive l'andamento della f nella direzione $\{x = x^* + \lambda d, \lambda \geq 0\}$. Consideriamo lo sviluppo di Taylor del primo ordine di h in $\lambda = 0$:

$$f(x^* + \lambda d) = h(x) = h(0) + \frac{dh}{d\lambda}\Big|_{\lambda=0} \cdot \lambda + O(\lambda) =$$

poiché $\frac{dh}{d\lambda}\Big|_{\lambda=0} = d^T \nabla f(x^*)$, si ha:

$$= f(x^*) + d^T \nabla f(x^*) \cdot \lambda + O(\lambda);$$

se $d^T \nabla f(x^*) < 0$, allora $\exists \bar{\lambda} > 0$ tale che $f(x^* + \lambda d) < f(x^*)$, per ogni $\lambda \in (0, \bar{\lambda}]$. In altri termini, è possibile sapere come decresce "localmente" la funzione obiettivo f; in effetti si è interessati a controllare l'andamento della f solamente in quelle direzioni che si mantengono all'interno dell'insieme ammissibile. Si considerino

perciò solo le direzioni cosiddette *ammissibili* in $x^* \in S$, ossia i vettori $d \in Z(x^*)$, dove:

$$Z(x^*) = \{d \in \mathbb{R}^n : x^* + \lambda d \in S, \lambda \in (0, \lambda^*]\}, \text{ per un certo } \lambda^* > 0. \tag{1.3}$$

Si può dimostrare il seguente teorema.

Teorema 1.2.1. *Sia f una funzione continuamente differenziabile su un aperto contenente $S \subset \mathbb{R}^n$. Se $x^* \in S$ è un minimo locale di f, allora*

$$d \in Z(x^*) \Rightarrow d^T \nabla f(x^*) \geq 0.$$

Un punto $x^* \in S$ tale che $d^T \nabla f(x^*) \geq 0 \ \forall d \in Z(x^*)$ è detto *punto stazionario* di f.

Se $x^* \in \overset{\circ}{S}$, essendo $\overset{\circ}{S}$ l'interno di S, allora ogni direzione è ammissibile, ossia $Z(x^*) = \mathbb{R}^n$. In questo caso, scegliendo $d_1 = \nabla f(x^*)$ e $d_2 = -\nabla f(x^*)$, dal teorema 1.2.1 discende la caratterizzazione classica del minimo locale.

Teorema 1.2.2. *Sia f una funzione continuamente differenziabile su un aperto contenente $S \subset \mathbb{R}^n$. Se $x^* \in S$ è un minimo locale di f interno a S, allora $\nabla f(x^*) = 0$.*

Il teorema 1.2.2 fornisce una condizione necessaria *del primo ordine* perché un punto dell'insieme ammissibile sia un minimo locale e vale in particolare nel caso di ottimizzazione non vincolata; infatti, in questo contesto, ogni punto di $S = \mathbb{R}^n$ è ovviamente un punto interno di $\mathbb{R}^n$.
Supponiamo ora che f sia almeno due volte differenziabile con derivata continua e indichiamo con $\nabla^2 f(x^*)$ la sua matrice hessiana; ricordiamo che una funzione $f : S \subset \mathbb{R}^n \to \mathbb{R}$ è convessa in $x^* \in S$ se e solo se $d^T \nabla^2 f(x^*) d \geq 0, \ \forall d \in Z(x^*)$, ossia se e solo se la matrice hessiana è semidefinita positiva. Una condizione necessaria del *secondo ordine* è la seguente.

Teorema 1.2.3. *Sia f una funzione due volte continuamente differenziabile definita su un aperto contenente $S \subset \mathbb{R}^n$. Se $x^* \in S$ è un minimo locale di f interno a S, allora $\nabla f(x^*) = 0$ e $d^T \nabla^2 f(x^*) d \geq 0$.*

Il seguente teorema fornisce invece una condizione sufficiente del secondo ordine perché un punto sia di minimo locale per una funzione.

Teorema 1.2.4. *Sia f una funzione due volte continuamente differenziabile definita su un aperto contenente $S \subset \mathbb{R}^n$. Se $\nabla f(x^*) = 0$ e $d^T \nabla^2 f(x^*) d > 0$ allora $x^* \in S$ è un minimo locale stretto di f.*

Se ci riferiamo in particolare a problemi di *programmazione convessa*, è possibile dimostrare che tutti i punti stazionari sono punti di minimo globale e tutti i minimi locali coincidono col minimo globale (cfr. [34]).

Teorema 1.2.5. *Se f è una funzione continuamente differenziabile e convessa su un aperto contenente S insieme convesso, allora $x^* \in S$ è un punto di minimo globale di f se e solo se x^* è un punto stazionario.*

Teorema 1.2.6. *Ogni punto di minimo locale di $f : S \to \mathbb{R}$ funzione convessa, con $S \subset \mathbb{R}^n$ convesso, è anche un punto di minimo globale per f.*

Dimostrazione. Sia x^* un punto di minimo locale di f. Si supponga per assurdo che esista $\bar{x} \in S$ tale che $f(\bar{x}) < f(x^*)$. La convessità di f implica che, per $0 < \lambda < 1$, si ha:

$$f(x^* + \lambda(\bar{x} - x^*)) = f(\lambda\bar{x} + (1-\lambda)x^*) \leq \lambda f(\bar{x}) + (1-\lambda)f(x^*) < f(x^*),$$

il che contraddice l'ipotesi che x^* sia un minimo locale di f; in tal caso infatti dovrebbe esistere $\lambda^* > 0$ tale che:

$$f(x^*) < f(x^* + \lambda(\bar{x} - x^*)), \quad \forall \lambda \in (0, \lambda^*].$$

□

La risoluzione di problemi di *programmazione concava* risulta in generale più complicata rispetto ai problemi convessi, in quanto questo tipo di problemi presenta un numero elevato di minimi locali non globali. Esistono comunque risultati importanti che consentono di avere informazioni sui punti di minimo globale quando la funzione obiettivo è concava (cfr. [34]).

Teorema 1.2.7. *Sia f una funzione concava a valori reali definita su un insieme convesso e compatto $S \subset \mathbb{R}^n$. Allora f raggiunge il minimo globale in punti che appartengono alla frontiera di S.*

Nel caso particolare in cui la funzione sia concava su un poliedro, è possibile dimostrare che la minimizzazione può essere effettuata solo sull'insieme discreto dei vertici di S, piuttosto che su tutta la frontiera dell'insieme ammissibile.

Se la funzione non è strettamente concava, il minimo globale può essere raggiunto anche in punti che non necessariamente sono punti estremi di S.

1.2.1 Condizioni di Kuhn-Tucker

Si è visto che i teoremi 1.2.2 e 1.2.3 rappresentano condizioni necessarie perché un punto sia di minimo locale quando la funzione obiettivo è definita su un insieme aperto; conseguentemente tali teoremi valgono per problemi di ottimizzazione non vincolata.
Se però l'insieme di definizione della f non è un insieme aperto di $\mathbb{R}^n$, l'individuazione dei minimi locali, e quindi del minimo globale, dipende sia dalla funzione f che dai vincoli $g_i,\ i = 1, \ldots, p$.

Nel contesto dell'ottimizzazione vincolata, riformuliamo la condizione necessaria per l'ottimalità locale del teorema 1.2.2 in termini di un sistema di equazioni e disequazioni, note come *condizioni di Kuhn-Tucker*. Queste condizioni, oltre che necessarie, sono condizioni sufficienti per l'ottimalità globale sotto opportune ipotesi di convessità della funzione obiettivo e dell'insieme ammissibile.

S descritto da vincoli di disuguaglianza

Inizialmente, prendiamo in considerazione il problema 1.1.1 in cui S è definito mediante soli vincoli di disuguaglianza:

Problema 1.2.1.

$$\begin{aligned} &\text{minimizzare} \quad f(x), \\ &x \in S, \end{aligned} \tag{1.4}$$

dove $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0,\ i = 1, \ldots, p\}$, $g_i : A \to \mathbb{R}$ e $A \supset S$ aperto di $\mathbb{R}^n$.

Siano f e g_i funzioni continuamente differenziabili su A. Per il teorema 1.2.1, un punto di minimo locale $x^* \in S$ soddisfa la condizione

$$d^T \nabla f(x^*) \geq 0,\ \forall d \in Z(x^*),$$

dove $Z(x^*)$ è l'insieme (o *cono*) delle direzioni ammissibili definito in (1.3).
Sia $I(x^*) := \{i \in \{1, \ldots, p\} : g_i(x^*) = 0\}$ l'insieme degli indici dei *vincoli attivi* in x^*. Il ragionamento utilizzato per ricavare il teorema 1.2.1 permette di osservare che:

$$d^T \nabla g_i(x^*) < 0 \Rightarrow g_i(x^* + \lambda d) < g_i(x^*),\ 0 < \lambda \leq \bar{\lambda}_i,\ \text{per un } \bar{\lambda}_i > 0,$$

che, nel caso di vincoli attivi, implica un allontanamento dal vincolo g_i nella direzione d, pur restando nell'insieme S, (d sarà allora una direzione ammissibile); analogamente

$$d^T \nabla g_i(x^*) > 0 \Rightarrow g_i(x^* + \lambda d) > g_i(x^*),\ 0 < \lambda \leq \tilde{\lambda}_i,\ \text{per un } \tilde{\lambda}_i > 0,$$

cioè, nel caso di vincoli attivi, si ha un allontanamento dal vincolo g_i e dall'insieme S, (d non sarà allora una direzione ammissibile).
D'altra parte, ogni vincolo non attivo in x^*, cioè tale che $g_i(x^*) < 0$, non determina alcuna limitazione sull'insieme delle direzioni ammissibili $Z(x^*)$, dato che, per la continuità di g_i, $g_i(x^*) < 0$ vale su tutto un intorno del punto. Allora, indicati con $l(x^*)$ e $L(x^*)$ gli insiemi:

$$l(x^*) := \{d \in \mathbb{R}^n : d^T \nabla g_i(x^*) < 0, i \in I(x^*)\},$$

e

$$L(x^*) := \{d \in \mathbb{R}^n : d^T \nabla g_i(x^*) \leq 0, i \in I(x^*)\},$$

da quanto detto segue che:

$$l(x^*) \subseteq Z(x^*) \subseteq L(x^*). \tag{1.5}$$

In particolare, è possibile dimostrare che nel caso di problemi linearmente vincolati, in cui cioè i vincoli sono del tipo $g_i(x) = a_i^T x - b_i$, $a_i \in \mathbb{R}^n \setminus \{0\}$, $b_i \in \mathbb{R}$, si ha

$Z(x^*) = L(x^*)$. D'altra parte, è possibile costruire esempi di problemi con vincoli non lineari per i quali $Z(x^*) = l(x^*)$.

Osserviamo inoltre che, essendo la chiusura $\overline{M}$ di un insieme $M \subseteq \mathbb{R}^n$ il più piccolo insieme chiuso contenente M ed essendo il prodotto scalare $d^T \nabla g_i(x^*)$ continuo, si ha:

$$d^T \nabla g_i(x^*) \leq 0,\ \forall d \in Z(x^*) \Rightarrow d^T \nabla g_i(x^*) \leq 0,\ \forall d \in \overline{Z(x^*)}. \tag{1.6}$$

In maniera analoga:

$$d^T \nabla f(x^*) \geq 0,\ \forall d \in Z(x^*) \Rightarrow d^T \nabla f(x^*) \geq 0,\ \forall d \in \overline{Z(x^*)}. \tag{1.7}$$

Consideriamo ora la seguente definizione.

Definizione 1.2.1. *I vincoli g_i sono detti* regolari *o* qualificati *in $x^* \in S$ quando*

$$\overline{Z(x^*)} = L(x^*).$$

Ogni condizione che assicura la regolarità di un vincolo è detta condizione di qualificazione.

Il seguente teorema fornisce degli esempi di condizioni di qualificazione (cfr. [34]).

Teorema 1.2.8. *Sia $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0,\ i = 1, \ldots, p\}$, con $g_i : A \to \mathbb{R}$ e $A \supseteq S$. I vincoli g_i sono qualificati in x se è verificata una delle seguenti condizioni:*

i) $g_i(x) = a_i^T x - b_i,\ a_i \in \mathbb{R}^n \setminus \{0\},\ b_i \in \mathbb{R},\ i = 1, \ldots, p$ (vincoli lineari);

ii) g_i è una funzione convessa $\forall\ i = 1, \ldots, p$ ed esiste un $\bar{x} \in S$ tale che $g_i(\bar{x}) < 0$ $\forall\ i = 1, \ldots, p$ (condizione di Slater);

iii) i vettori $\nabla g_i(x^*),\ i \in I(x^*)$, sono linearmente indipendenti.

Le condizioni i) e ii) assicurano la regolarità in ogni $x^* \in S$, mentre la iii) richiede la conoscenza di x^*.

Ricordando le (1.6) e (1.7) e considerato che se i vincoli sono qualificati vale la definizione 1.2.1, dal teorema 1.2.1 si ottiene la seguente condizione necessaria di ottimalità.

Teorema 1.2.9. *Siano f e $g_i,\ i = 1, \ldots, p$, funzioni differenziabili su un aperto contenente S. Sia x^* un punto di minimo locale per f e i vincoli g_i siano qualificati in x^*. Allora*

$$d^T \nabla g_i(x^*) \leq 0,\ \forall\ i \in I(x^*) \Rightarrow d^T \nabla f(x^*) \geq 0.$$

Ricordiamo l'enunciato del lemma di Farkas [21].

Lemma 1.2.1 (Farkas). *Data una matrice $A \in \mathbb{R}^{m \times n}$ e un vettore $q \in \mathbb{R}^n$, la relazione*

$$x^T q \geq 0,\ \forall x \in \mathbb{R}^n : x^T A \leq 0,$$

vale se e solo se esiste $u \in \mathbb{R}^m_+$ tale che

$$u^T A = q.$$

Applichiamo opportunamente il lemma 1.2.1 al nostro contesto:
se $d^T \nabla f(x^*) \geq 0,\ \forall d \in \mathbb{R}^n : d^T \nabla g_i(x^*) \leq 0,\ i \in I(x^*)$, allora esistono delle costanti $\lambda_i \geq 0, i \in I(x^*)$, tali che

$$\nabla f(x^*) + \sum_{i \in I(x^*)} \lambda_i \nabla g_i(x^*) = 0. \tag{1.8}$$

Poiché l'insieme $I(x^*)$ non è noto a priori, formuliamo l'equazione (1.8) in modo equivalente, come indicato nel teorema successivo, che rappresenta la riformulazione della condizione necessaria di ottimalità locale del teorema 1.2.2 per il caso vincolato (cfr. [34]).

Teorema 1.2.10 (Condizioni necessarie di Kuhn-Tucker del primo ordine). *Siano f e $g_i,\ i = 1, \ldots, p$, funzioni continuamente differenziabili su un aperto contenente S e sia x^* un punto di minimo locale per f tale che i vincoli g_i siano qualificati in x^*. Allora valgono le seguenti condizioni di Kuhn-Tucker:*

i) $g_i(x^*) \leq 0,\ i = 1, \ldots, p$;

ii) $\exists \lambda_i \geq 0$ *tale che* $\lambda_i g_i(x^*) = 0,\ i = 1, \ldots, p$;

iii) $\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*) = 0.$

Se le funzioni coinvolte nella minimizzazione sono convesse, le condizioni di Kuhn-Tucker del teorema 1.2.10 sono anche sufficienti per l'individuazione di un minimo *globale* della funzione, come dimostra il risultato seguente (cfr. [34]).

Teorema 1.2.11 (Condizioni sufficienti di Kuhn-Tucker del primo ordine). *Siano f e $g_i,\ i = 1, \ldots, p$, funzioni continuamente differenziabili e convesse su un aperto contenente S. Se in $x^* \in S$ valgono le condizioni di Kuhn-Tucker del teorema 1.2.10, allora x^* è un punto di minimo globale per f in S.*

Si osservi che il teorema 1.2.11 non richiede che i vincoli siano qualificati.

S descritto da vincoli di uguaglianza e disuguaglianza

Consideriamo ora il problema 1.1.1 in cui S sia descritto mediante vincoli di uguaglianza e disuguaglianza:

Problema 1.2.2.

$$\begin{aligned} &\text{minimizzare} \quad f(x), \\ &x \in S, \end{aligned} \tag{1.9}$$

dove $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0,\ (i = 1, \ldots, p),\ h_j(x) = 0,\ (j = 1, \ldots, t)\}$.

In questo caso, il teorema 1.2.10 può essere riformulato nel modo seguente (cfr. [34]).

Teorema 1.2.12 (Condizioni necessarie di Kuhn-Tucker del primo ordine). *Siano f, g_i, $(i = 1, \dots, p)$, e h_j, $(j = 1, \dots, t)$ funzioni continuamente differenziabili su un aperto contenente S e sia x^* un punto di minimo locale per f. Inoltre i vettori $\nabla g_i(x^*)$, $i \in I(x^*)$ e $\nabla h_j(x^*)$, $j = 1, \dots, t$ siano linearmente indipendenti. Allora valgono le seguenti:*

i) $g_i(x^*) \leq 0$, $i = 1, \dots, p$ *e* $h_j(x^*) = 0$, $j = 1, \dots, t$;

ii) $\exists \lambda_i \geq 0$, $(i = 1, \dots, p)$ *e* $\mu_j \in \mathbb{R}$, $(j = 1, \dots, t)$ *tali che*

$$\nabla f(x^*) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*) + \sum_{j=1}^{t} \mu_j \nabla h_j(x^*) = 0,$$

$$\lambda_i g_i(x^*) = 0, \quad i = 1, \dots, p.$$

Il punto i) del teorema garantisce che il punto di minimo locale x^* appartenga al dominio S, mentre il punto ii) non è altro che la riformulazione della condizione $\nabla f(x^*) = 0$ del teorema 1.2.2 per il problema 1.2.2.
Si osservi inoltre che, similmente al caso di vincoli di sola disuguaglianza, se le funzioni f e g_i sono convesse e se le h_j sono affini, allora le condizioni i) e ii) sono condizioni sufficienti affinché x^* sia un minimo globale, non solo locale, per f.

1.2.2 Concetto di Lagrangiana e condizioni del secondo ordine

Introduciamo ora il concetto di *Lagrangiana*. Consideriamo il problema 1.2.2, dove

$$g : \mathbb{R}^n \to \mathbb{R}^p, \quad g(x) = (g_1(x), \dots, g_p(x))^T,$$

$$h : \mathbb{R}^n \to \mathbb{R}^t, \quad h(x) = (h_1(x), \dots, h_t(x))^T.$$

Sia inoltre $\lambda = (\lambda_1, \dots, \lambda_p)^T$ e $\mu = (\mu_1, \dots, \mu_t)^T$.

Definizione 1.2.2. *La funzione*

$$L(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x)$$

è chiamata Lagrangiana *associata al problema 1.2.2.*

In termini di Lagrangiana, il teorema 1.2.12 assume la forma seguente (cfr. [34]):

Teorema 1.2.13 (Condizioni necessarie di Kuhn-Tucker del primo ordine). *Siano f, g_i, $(i = 1, \dots, p)$, e h_j, $(j = 1, \dots, t)$ funzioni continuamente differenziabili su un aperto contenente S e sia x^* un punto di minimo locale per f. Inoltre i vettori $\nabla g_i(x^*)$, $i \in I(x^*)$ e $\nabla h_j(x^*)$, $j = 1, \dots, t$ siano linearmente indipendenti. Allora esistono $\lambda \in \mathbb{R}^p$, $\lambda \geq 0$ e $\mu \in \mathbb{R}^t$ tali che:*

i) $\nabla_\lambda L(x^*, \lambda, \mu) \leq 0$ *e* $\nabla_\mu L(x^*, \lambda, \mu) = 0$;

ii) $\nabla_x L(x^*, \lambda, \mu) = 0$ *e* $\lambda^T g(x^*) = 0$.

I vettori λ e μ sono detti *moltiplicatori di Kuhn-Tucker*. I punti di $\mathbb{R}^n$ che soddisfano le condizioni i) e ii) del teorema 1.2.13 sono detti *punti di Kuhn-Tucker* e rappresentano l'analogo, nel caso vincolato, dei punti stazionari per problemi non vincolati.

Siano ora f, g e h funzioni due volte continuamente differenziabili in un intorno di un punto di minimo locale x^* e valga la condizione di qualificazione dei vincoli considerata nel teorema 1.2.13. In questo caso, oltre alle condizioni necessarie già viste, è possibile dimostrare che la matrice hessiana $\nabla_x^2 L(x^*, \lambda, \mu)$ è semidefinita positiva rispetto allo *spazio tangente*

$$T(x^*) = \{y \in \mathbb{R}^n : \ y^T \nabla g_i(x^*) = 0, \ i \in I(x^*), \ y^T \nabla h_j(x^*) = 0, \ j = 1, \ldots, t\}.$$

Vale infatti il seguente risultato (cfr. [34]).

Teorema 1.2.14 (Condizioni necessarie di Kuhn-Tucker del secondo ordine). *Siano* f, $g_i, (i = 1, \ldots, p)$, *e* h_j, $(j = 1, \ldots, t)$ *funzioni due volte continuamente differenziabili su un aperto contenente* S *e sia* x^* *un punto di minimo locale per* f. *Inoltre i vettori* $\nabla g_i(x^*)$, $i \in I(x^*)$ *e* $\nabla h_j(x^*)$, $j = 1, \ldots, t$ *siano linearmente indipendenti. Allora esistono* $\lambda \in \mathbb{R}^p$, $\lambda \geq 0$ *e* $\mu \in \mathbb{R}^t$ *tali che:*

i) $\nabla_\lambda L(x^*, \lambda, \mu) \leq 0$ *e* $\nabla_\mu L(x^*, \lambda, \mu) = 0$;

ii) $\nabla_x L(x^*, \lambda, \mu) = 0$ *e* $\lambda^T g(x^*) = 0$;

iii) $y^T \nabla_x^2 L(x^*, \lambda, \mu) y \geq 0, \ \forall y \in T(x^*)$.

Le condizioni di ottimalità del secondo ordine risultano particolarmente interessanti per problemi di ottimizzazione non convessa, in quanto consentono di mettere in relazione i vincoli attivi del problema con gli autovalori della matrice hessiana $\nabla_x^2 L(x^*, \lambda, \mu)$ (cfr. [34]).

1.3 Condizioni di ottimalità globale

Non sempre il problema 1.1.1 ammette soluzione; il primo passo nello studio di un problema di ottimizzazione globale dunque è quello di stabilire quando tale problema è *ben posto*. Ricordiamo che un problema numerico si dice *ben posto* quando ammette un'unica soluzione che dipende con continuità dai dati.
Una condizione sufficiente (ma non necessaria) per l'esistenza del minimo globale è rappresentata dal ben noto teorema di Weierstrass.

Teorema 1.3.1 (Weierstrass). *Sia* $S \subset \mathbb{R}^n$ *un insieme compatto non vuoto e sia* $f : S \to \mathbb{R}$ *una funzione continua. Allora esiste almeno un punto di minimo globale di* f *in* S.

Il teorema 1.3.1 continua a valere se la funzione obiettivo è semicontinua inferiormente (cfr. [34]). Si ricordi, a tal proposito, la seguente definizione:

Definizione 1.3.1. *Una funzione a valori reali f definita su un insieme $S \subset \mathbb{R}^n$ è detta* semicontinua inferiormente *(l.s.c.) in un punto $x \in S$ se, per ogni successione $\{x_i\}_{i\in\mathbb{N}}$, $x_i \in S$, tale che $\lim_{i\to\infty} x_i = x$ e tale che esista $\lim_{i\to\infty} f(x_i)$, si ha $f(x) \leq \lim_{i\to\infty} f(x_i)$.*

Questa proprietà viene spesso rappresentata come $f(x) = \lim_{y\to x} \inf f(y)$. Similmente si definisce la *semicontinuità superiore* (u.s.c.) di f in x, indicata come $f(x) = \lim_{y\to x} \sup f(y)$. Si osservi che una funzione è continua se è semicontinua sia superiormente che inferiormente.

Vale il seguente teorema.

Teorema 1.3.2. *Sia $S \subset \mathbb{R}^n$ un insieme compatto non vuoto; sia $f : S \to \mathbb{R}$ una funzione semicontinua inferiormente. Allora esiste almeno un punto di minimo globale di f in S.*

Il teorema di Weierstrass garantisce l'esistenza del minimo globale quando la funzione è almeno semicontinua su un insieme compatto, quindi per problemi di ottimizzazione vincolata. È possibile ottenere delle condizioni di esistenza simili anche per il caso di problemi di ottimizzazione su insiemi non limitati, ad esempio individuando sottoinsiemi opportuni di S che contengano il minimo globale.
A questo scopo, diamo la seguente definizione.

Definizione 1.3.2. *Ogni insieme non vuoto*

$$L(f;\alpha) = \{x \in S : f(x) \leq \alpha,\ \alpha \in \mathbb{R}\},$$

è detto insieme di livello *della funzione f.*

L'insieme $L(f;\alpha)$ può anche essere scritto nella forma:

$$L_0(f) = \{x \in S : f(x) \leq f(x_0),\ x_0 \in S\}.$$

Dal teorema 1.3.1 discende il seguente teorema (cfr. [34]).

Teorema 1.3.3. *Sia $f : \mathbb{R}^n \to \mathbb{R}$ una funzione continua. Se esiste un insieme di livello di f non vuoto e compatto, allora f possiede almeno un punto di minimo globale in $\mathbb{R}^n$.*

Infatti, se un insieme di livello $L_0(f)$ è compatto, per il teorema 1.3.1 e per definizione di insieme di livello, f raggiunge il minimo globale in $L_0(f)$.

Un'altra condizione di esistenza del minimo globale in $\mathbb{R}^n$ può essere stabilita quando la funzione obiettivo è coerciva.

Definizione 1.3.3. *Una funzione f definita in $\mathbb{R}^n$ è* coerciva *se:*

$$\lim_{\|x\|\to\infty} f(x) = +\infty.$$

Vale il seguente teorema (cfr. [34]).

Teorema 1.3.4. *Sia $f : \mathbb{R}^n \to \mathbb{R}$ una funzione continua e coerciva. Allora f possiede almeno un punto di minimo globale in $\mathbb{R}^n$.*

Dimostrazione. Sia $x_0 \in \mathbb{R}^n$; per la coercività di f esiste una costante $M > 0$ tale che, $\forall x \in \mathbb{R}^n$ con $\|x\| > M$, sia $f(x) \geq f(x_0)$. Allora il problema originale

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

si riduce al problema

$$\min_{x\in S} f(x),$$

dove $S = \{x : \|x\| \leq M\} \subset \mathbb{R}^n$ è compatto. Per il teorema 1.3.1, f ha almeno un punto di minimo globale in S, dunque anche in $\mathbb{R}^n$. □

Una condizione più generale di esistenza (solo sufficiente) si ottiene dal teorema di Weierstrass considerando ancora gli insiemi di livello della f piuttosto che tutti i punti di S.

Teorema 1.3.5. *Sia $f : S \to \mathbb{R}$ una funzione continua su S non vuoto e chiuso. Se esiste un insieme di livello di f limitato, allora f possiede almeno un punto di minimo globale in S.*

Dimostrazione. Essendo f continua ed S chiuso, ogni insieme di livello $L(f;\alpha)$ è anch'esso chiuso e quindi, per l'ipotesi fatta, compatto. Dal teorema di Weierstrass discende che f ha minimo in $L(f;\alpha)$ e tale minimo è anche globale, per definizione di insieme di livello. □

Si è visto che, in opportune condizioni, il problema di individuare il minimo globale f^* della funzione obiettivo f è ben posto. Al contrario, il problema di determinare un punto di minimo globale $x^* \in S^*$ per f è un problema *mal posto*: infatti x^* può non essere unico; inoltre esistono funzioni continue simili tra loro rispetto ad una particolare "distanza" (nel senso che la differenza massima assoluta tra corrispondenti valori delle funzioni è arbitrariamente piccola), aventi però punti di minimo globale molto distanti tra loro (cfr. [62]). Questo significa che trovare un punto di minimo globale x^* per f è più complicato che individuare l'ottimo assoluto f^*.

1.4 Caratterizzazione del minimo globale

Dato il problema di ottimizzazione globale 1.1.1, si vorrebbero trovare delle caratterizzazioni generali del minimo globale, ossia condizioni necessarie e sufficienti perché un punto dell'insieme ammissibile sia un minimo globale, simili ad esempio al criterio del gradiente nullo per i minimi locali. In effetti *una caratterizzazione generale del minimo globale non esiste*, anche se recentemente sono stati ottenuti diversi risultati teorici in questa direzione.

Un primo risultato si ottiene quando l'insieme ammissibile S è un insieme *robusto*, ossia è tale che $S = Cl(\overset{\circ}{S})$ (S coincide con la chiusura del suo interno) (cfr. [71]).

Teorema 1.4.1 (Zheng, 1985). *Sia f una funzione continua a valori reali definita su un insieme robusto $S \subset \mathbb{R}^n$. Allora $x^* \in S$ è un punto di minimo globale della f su S se e solo se l'insieme di livello $L(f; f(x^*))$ è tale che $mis(L(f; f(x^*))) = 0$, dove* mis *indica la misura di Lebesgue di un insieme.*

Un risultato molto interessante per la caratterizzazione del minimo globale deriva dal tentativo di trasformare il problema dell'ottimizzazione globale in un problema di tipo locale (Hiriart-Urruty, 1998, [33]).

Preliminarmente, è necessario dare alcune definizioni.

Definizione 1.4.1. *L'insieme di $\mathbb{R}^{n+1}$*

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

è detto epigrafo *della funzione f.*

È possibile dimostrare che la semicontinuità inferiore di f in S è equivalente alla chiusura di $L(f; \alpha),\ \forall \alpha \in \mathbb{R}$ e alla chiusura di $epi(f)$ in $\mathbb{R}^{n+1}$.

Definizione 1.4.2. *Sia $f : S \to \mathbb{R}$ una funzione semicontinua inferiormente, con $S \subset \mathbb{R}^n$ insieme convesso non vuoto. Si definisce* involucro convesso *di $f(x)$ su S la funzione $F(x)$ che soddisfa le seguenti proprietà:*

i) $F(x)$ *è convessa su* S;

ii) $F(x) \leq f(x), \quad \forall x \in S$;

iii) se $h(x)$ è una funzione convessa definita su S tale che $h(x) \leq f(x)$, $\forall x \in S$, allora $h(x) \leq F(x),\ \forall x \in S$.

L'involucro convesso di una funzione è dunque la migliore sottostima convessa della $f(x)$ su S; inoltre $F(x)$ risulta semicontinua inferiormente.

Si consideri il seguente teorema.

Teorema 1.4.2. *Si consideri il problema:*

$$\min_{x\in S} f(x),$$

dove f è continua ed S è un insieme compatto e convesso di $\mathbb{R}^n$. Sia $F(x)$ l'involucro convesso di f su S. Allora:

$$f^* := \min\{f(x) : x \in S\} = \min\{F(x) : x \in S\}$$

e

$$\{y \in S : f(y) = f^*\} \subseteq \{y \in S : F(y) = f^*\}.$$

Il teorema 1.4.2 vale anche quando $f : \mathbb{R}^n \to \mathbb{R}$ è una funzione semicontinua inferiormente e coerciva in $\mathbb{R}^n$.

Dal teorema sopra citato discende che *ad ogni problema di ottimizzazione non convessa con insieme ammissibile convesso, può essere associato un problema di ottimizzazione convessa avente lo stesso minimo locale del problema iniziale.* Quindi, piuttosto che risolvere il problema iniziale, nelle condizioni ipotizzate dal teorema è preferibile calcolare il minimo dell'involucro convesso della funzione obiettivo.

In generale, individuare l'involucro convesso di una funzione non è molto più semplice che risolvere il problema dell'ottimizzazione globale della f. Esistono comunque dei casi particolari per i quali l'involucro convesso si può rappresentare mediante la funzione obiettivo e l'utilizzo di insiemi convessi. Si considerino ad esempio i seguenti risultati (cfr. [34]).

Teorema 1.4.3. *Siano $v_1, \ldots, v_k$ i vertici di un poliedro P e sia f una funzione concava su P. Allora l'involucro convesso di $f(x)$ su P è dato da:*

$$F(x) = \min_{\alpha} \sum_{i=1}^{k} \alpha_i f(v_i),$$

essendo

$$\sum_{i=1}^{k} \alpha_i v_i = x, \quad \sum_{i=1}^{k} \alpha_i = 1, \ \alpha_i \geq 0, \quad i = 1, \ldots, k,$$

con $\alpha = (\alpha_1, \ldots, \alpha_k)$.

Teorema 1.4.4. *Sia S un simplesso di vertici $v_0, v_1, \ldots, v_n$ e sia f una funzione concava definita su S. Allora l'involucro convesso di $f(x)$ su S è rappresentato dalla funzione $F(x) = c^T x + b$, univocamente individuata dal sistema di equazioni lineari*

$$f(v_i) = c^T v_i + b, \quad i = 0, \ldots, n.$$

La definizione di involucro convesso è utile per caratterizzare il minimo globale di una funzione differenziabile; se il teorema 1.2.2 fornisce una condizione necessaria perché un punto di S sia un punto di minimo globale, il seguente teorema rappresenta una condizione necessaria e sufficiente perché un minimo sia globale (cfr. [33]).

Teorema 1.4.5. *Se $f : \mathbb{R}^n \to \mathbb{R}$ una funzione differenziabile in $\mathbb{R}^n$. Allora x^* è un minimo globale di f in $\mathbb{R}^n$ se e solo se:*

i) $\nabla f(x^*) = 0$;

ii) $F(x^*) = f(x^*)$.

In questo caso, F è differenziabile in x^* e $\nabla F(x^*) = 0$.

La proprietà ii) è ciò che consente di stabilire se un punto stazionario è anche un punto di minimo globale. Dal teorema 1.4.5 discende, per esempio, che le funzioni differenziabili i cui punti stazionari sono minimi globali sono quelle funzioni per cui

$$\{x^* |\ \nabla f(x^*) = 0\} \subset \{x^* |\ F(x^*) = f(x^*)\}.$$

Poiché, come già osservato, il calcolo dell'involucro convesso di f in un punto non è semplice, la condizione ii) non è applicabile direttamente; la sua negazione però può essere utile per individuare punti che non sono di minimo globale.

Proposizione 1.4.1. *Un punto $\bar{x}$, stazionario o meno, per il quale $l < f(\bar{x})$, con l stima superiore di $F(\bar{x})$, non può essere un minimo globale di f.*

Per concludere, consideriamo problemi con funzione obiettivo continua su un insieme compatto. In questo contesto così generale non esistono condizioni di ottimalità applicabili; è però possibile rendere in un certo senso *convesso* il problema mediante l'integrazione della funzione obiettivo.
Sia S un insieme aperto limitato non vuoto e $f : S \to \mathbb{R}$ sia una funzione continua. Un risultato che probabilmente risale ai tempi di Laplace e attualmente è utilizzato in Probabilità e Statistica nella teoria delle *grandi deviazioni*, è il seguente:

$$\min_{x \in S} f(x) = \lim_{k \to +\infty} \Big[\frac{1}{k} \mathrm{Log} \int_S e^{-kf(x)} dx \Big]. \tag{1.10}$$

Il problema che si pone ora è capire come si può approssimare in modo efficiente l'integrale della (1.10) e stabilire se la procedura utilizzata risulta stabile per $k \to +\infty$. Se g è una funzione continua strettamente positiva su S, mediante la trasformazione $f = \mathrm{Log}(g)$ in (1.10) si può osservare che il comportamento della successione

$$r_k(\bar{x}) = \int_S \left(\frac{g(\bar{x})}{g(x)} \right)^k dx,\ \bar{x} \in S, \tag{1.11}$$

è strettamente correlato col problema di capire se la differenza $g(\bar{x}) - g(x)$ (o $f(\bar{x}) - f(x)$) è negativa $\forall x \in S$ (cioè $\bar{x}$ è un punto di minimo globale). In effetti è possibile dimostrare che *$\bar{x}$ è un punto di minimo globale di g in S se e solo se la successione $r_k(\bar{x})$ è limitata* (Falk, 1973, [33]). Dalla relazione intercorrente tra g ed f segue che $\bar{x}$ è punto di minimo globale anche per f.

Un risultato simile che riguarda una successione di approssimazioni dell'unico punto di minimo globale di una funzione f su S è il seguente (cfr. [72]).

Teorema 1.4.6 (Pincus, 1968). *Sia $f : S \to \mathbb{R}$ una funzione continua ed S la chiusura di un insieme aperto limitato non vuoto. Si supponga inoltre che f possieda un unico punto di minimo globale x_{min} in S. Allora la successione*

$$x_k := \frac{\int_S x e^{-kf(x)} dx}{\int_S e^{-kf(x)} dx}, \ k = 1, 2, \ldots$$

converge a x_{min} per $k \to +\infty$.

Alla luce delle problematiche emerse dalla precedente trattazione, nel capitolo successivo verranno introdotte le caratteristiche generali dei metodi usualmente usati per la risoluzione, sia locale che globale, del problema 1.1.1.

Capitolo 2

Proprietà generali

Oggetto del nostro studio sarà la risoluzione del problema dell'ottimizzazione globale definito dalla (1.1) nell'ipotesi che f sia una funzione *continua* su un insieme S *compatto*.
Per comodità indicheremo con f^* il minimo globale della f e con S^* l'insieme di tutti i suoi punti di minimo globale, ossia l'insieme:

$$S^* = \{x \in S : f(x) = f^*\}.$$

In generale S^* può essere un insieme finito, numerabile, non numerabile o di volume positivo in S; per la trattabilità del problema dal punto di vista algoritmico, supporremo S^* al più numerabile.

2.1 Ottimizzazione locale

L'ottimizzazione locale riveste una grande importanza nella risoluzione del problema dell'ottimizzazione globale. È pertanto opportuno introdurre i concetti usati più frequentemente in questo contesto.

Innanzitutto osserviamo che in genere una procedura numerica non è in grado di fornire una soluzione esatta del problema 1.1.1; pertanto, almeno per il momento, supporremo di aver raggiunto un generico minimo locale $f_i^* = f(x_i^*)$ quando viene individuato un punto nell'insieme:

$$B_i = \{x \in S \ : \ f(x) \leq f_i^* + \varepsilon, \ \|x - x_i^*\| \leq \delta\},$$

di misura $mis(B_i) > 0$, con ε e δ costanti positive. Nonostante la misura di B_i sia positiva, normalmente l'individuazione di f_i^* o di x_i^* è legata a quella di insiemi più

ampi, quali la *regione di attrazione* e il *bacino* del minimo locale. Consideriamo preliminarmente le seguenti definizioni.

Definizione 2.1.1 (Algoritmo di minimizzazione locale). *Un algoritmo L è detto di* minimizzazione locale *per la funzione f se determina un punto di stazionarietà o di minimo locale secondo il seguente schema:*

$$x_{k+1} = x_k + \alpha_k p_k,$$

con $p_k \in \mathbb{R}^n$, $\|p_k\| = 1$, $\alpha_k \in \mathbb{R}^+$ e $-\nabla f(x_k)^T \cdot p_k > 0$.

Definizione 2.1.2 (Algoritmo di discesa). *Si definisce* algoritmo di discesa *per la funzione $f(x)$ un algoritmo di minimizzazione L che, partendo da $x \in S$, genera una successione $\{x_k\}$ tale che:*

i) $f(x_k + \beta p_k) \leq f(x_k + \alpha p_k)$, $\forall k = 1, 2, \ldots$ *e* $\forall \alpha, \beta \in \mathbb{R}^+$ *con* $0 \leq \alpha < \beta \leq \alpha_k$;

ii) $x_k \underset{k\to\infty}{\longrightarrow} \bar{x}$, *con $\bar{x}$ punto di stazionarietà per f e $\{x_k\}$ contenuta in S.*

Un particolare algoritmo di discesa è il *metodo di più ripida discesa* (metodo *steepest descent*) secondo cui $p_k = -\nabla f(x_k)/\|\nabla f(x_k)\|_2$; in questo caso p_k prende il nome di *direzione di più ripida discesa.*

La regione di attrazione e il bacino di un minimo locale si definiscono come segue.

Definizione 2.1.3 (Regione di attrazione). *Si definisce* regione di attrazione *di un minimo locale f_i^* relativa ad un algoritmo di discesa A, il più grande sottoinsieme $Attr(f_i^*)$ di S tale che, applicando A ad un qualunque punto $x \in Attr(f_i^*)$, si ottiene f_i^*.*

Se esistono diversi punti di minimo locale $x_{i_j}^*$ che conducono allo stesso minimo locale f_i^*, allora la regione di attrazione di f_i^* è data dall'unione di altrettanti insiemi connessi disgiunti.

Definizione 2.1.4 (Bacino di un minimo). *Si definisce* bacino *di un minimo locale f_i^* corrispondente ad un singolo insieme connesso della regione di attrazione, il più grande insieme di livello contenuto nella regione di attrazione stessa.*

2.1.1 Importanza delle regioni di attrazione

Alla regione di attrazione di ogni minimo locale f_i^* è possibile associare un numero $p_i > 0$ che rappresenta la probabilità di raggiungere B_i, ossia un'approssimazione di f_i^*, mediante l'applicazione di un algoritmo di discesa a partire da un qualunque punto scelto a caso in S.
Supponendo che la funzione obiettivo f possegga k minimi locali differenti:

$$f_1^* \leq f_2^* \leq \ldots \leq f_k^*,$$

allora si definiscono queste probabilità come:

$$p_i = \frac{mis(Attr(f_i^*))}{mis(S)}, \quad i = 1, \ldots, k.$$

Per un problema di ottimizzazione globale è fondamentale il valore assunto da p_1. Ad esempio, se $p_1 = 1$, f è unimodale, ossia possiede un solo minimo, oppure tutti i minimi hanno lo stesso valore f^*. Il problema si riduce pertanto ad una questione di ottimizzazione locale. Se p_1 ha un valore prossimo ad uno, la maggior parte degli algoritmi sono in grado di trovare la soluzione ottima del problema, mentre se p_1 è vicino a zero l'individuazione del minimo globale si presenta estremamente difficile. In questo caso il procedimento richiederebbe la scelta di un'accuratezza molto piccola, compatibilmente con l'affidabilità dell'algoritmo.
Per funzioni abbastanza "lisce", in generale ci si attende che le regioni di attrazione siano tanto più ampie quanto più basso è il valore del minimo, cioè che $\max_i p_i = p_1$. In questa situazione il minimo globale è il più facile da determinare; raramente però le applicazioni reali soddisfano questa condizione.

2.1.2 Convergenza

Per definizione, i metodi di minimizzazione locale sono iterativi, cioè, a partire da un punto iniziale x_0, generano una successione di punti $\{x_k\}$, $x_k \in \mathbb{R}^n$, $k = 1, 2, \ldots$, che deve convergere ad una soluzione locale x^* del problema (cfr. [13]).

Definizione 2.1.5 (Convergenza). *Sia $x_k \in \mathbb{R}^n, k = 0, 1, \ldots$. Si dice che la successione $\{x_k\}$ converge ad un punto di minimo locale $x^* \in \mathbb{R}^n$ se, $\forall i$, la i-esima componente $(x_k - x^*)_i$ soddisfa la seguente relazione:*

$$\lim_{k \to \infty} (x_k - x^*)_i = 0.$$

Un metodo di minimizzazione locale è detto *localmente convergente* quando converge ad un punto di minimo locale x^* se applicato ad un punto sufficientemente vicino ad x^* mentre è detto *globalmente convergente* se tale proprietà di convergenza vale qualunque sia il punto di partenza in S.

L'efficienza di un metodo di minimizzazione locale può essere valutata in base alla sua *velocità di convergenza* (cfr. [63]).

Definizione 2.1.6 (Velocità di convergenza). *Si consideri una norma $\|\cdot\|$ in $\mathbb{R}^n$. Se esistono $K \geq 0$ e $\alpha \in [0, 1)$ tali che, $\forall k \geq K$,*

$$\|x_{k+1} - x^*\| \leq \alpha \|x_k - x^*\|,$$

allora si dice che la successione $\{x_k\}$ converge q-linearmente *a x^* secondo la norma $\|\cdot\|$.*
Se esiste una successione di scalari $\{\alpha_k\}$ convergente a zero tale che:

$$\|x_{k+1} - x^*\| \leq \alpha_k \|x_k - x^*\|,$$

allora si dice che la successione $\{x_k\}$ *converge* q-superlinearmente *a* x^*.
Infine, se esistono $K \geq 0$, $p > 0$ *e* $\alpha \geq 0$ *tali che,* $\forall k \geq K$,

$$\|x_{k+1} - x^*\| \leq \alpha \|x_k - x^*\|^p,$$

allora si dice che la successione $\{x_k\}$ *converge a* x^* *con* q-ordine *almeno* p. *Se* $p = 2$ *la convergenza è detta* q-quadratica.

Lo scalare α prende il nome di *tasso di convergenza.*

Si noti che la convergenza di q-ordine 1 è differente da quella lineare per la scelta diversa di α. Inoltre la convergenza lineare dipende dalla norma scelta mentre la convergenza q-superlineare e quella di q-ordine $p > 1$ sono indipendenti dalla norma di $\mathbb{R}^n$ utilizzata.

Nella maggior parte dei casi, i metodi di minimizzazione per funzioni continuamente differenziabili convergono localmente ad una soluzione locale x^* con velocità q-superlineare o q-quadratica. Dal punto di vista pratico, la convergenza q-quadratica è la più veloce dato che, una volta che x_k è prossimo a x^*, il numero di cifre significative di x_k uguali a quelle di x^* viene elevato al quadrato ad ogni iterazione. I metodi localmente q-superlineari sono anch'essi abbastanza veloci. La convergenza q-lineare invece può essere anche molto lenta specialmente quando la costante α, che in genere dipende dal problema, è vicina a 1. Conseguentemente, l'applicazione pratica di questi metodi viene evitata quando possibile, a meno che non sia noto che il tasso di convergenza è piccolo.

Dal punto di vista teorico è semplice definire velocità di convergenza superiori a quella quadratica, ad esempio quella cubica, ma dal punto di vista pratico queste velocità di convergenza non vengono utilizzate per problemi di ottimizzazione in più variabili.

Osservazione 2.1.1. Il prefisso "q" nei termini "lineare", "superlineare", "quadratico" etc. sta ad indicare una velocità di convergenza di tipo *quoziente,* per distinguerla da una forma più debole di convergenza, la cosiddetta *r-convergenza* (r=radice).
Una successione è *r-linearmente convergente* se l'errore $\|x_k - x^*\|$ è limitato superiormente da una successione di scalari $\{b_k\}$ che converge q-linearmente a zero.
Un tipico esempio di metodo r-lineare è l'algoritmo di bisezione per la risoluzione di equazioni non lineari.
Nel seguito della trattazione, il prefisso "q" verrà omesso intendendo sempre come velocità di convergenza, salvo diversa precisazione, quella di tipo "quoziente".

Esempi di metodi di minimizzazione (locale) localmente convergenti sono il metodo BFGS e il metodo di Newton, con velocità di convergenza superlineare e quadratica rispettivamente.

L'idea di base per la costruzione di metodi globalmente convergenti è in genere quella di sfruttare la rapida convergenza locale di procedure come il metodo di Newton, cercando inizialmente di avvicinarsi ad una regione dell'insieme ammissibile in cui è possibile applicare tali procedure, ad esempio mediante direzioni

di discesa. Esempi di metodi globalmente convergenti sono il metodo di più ripida discesa, i metodi di ricerca lineare (*line search methods*) e i metodi a regione di confidenza (*trust region methods*); il primo ha una velocità di convergenza di tipo lineare, gli ultimi due hanno velocità superlineare e quadratica rispettivamente, ma sotto opportune ipotesi (cfr. [13]). I metodi di ricerca lineare e a regione di confidenza sono molto utilizzati come procedure locali in moderni software come MATLAB®[1].

2.2 Ottimizzazione globale

Le difficoltà nella risoluzione del problema 1.1.1 dipendono:

i) dalla funzione da minimizzare, in particolare dallo sforzo computazionale necessario per ogni valutazione di funzione;

ii) dal numero dei minimi locali: maggiore è il numero dei punti di minimo locale x_i^* di una funzione e minori sono le misure delle rispettive regioni di attrazione, quindi la probabilità p_1 di individuare il minimo globale;

iii) dalla distribuzione dei minimi locali: se i minimi locali sono concentrati in una particolare regione dell'insieme ammissibile, individuare anche uno solo dei punti di minimo può portare all'individuazione di altri minimi vicini; sfruttare questo fatto normalmente è impossibile se i minimi sono distribuiti in S;

iv) dalla dimensione dello spazio di ricerca: è possibile dimostrare che lo sforzo computazionale necessario per generare e memorizzare i punti di prova cresce in modo esponenziale al crescere della dimensione; inoltre, in genere al crescere della dimensione aumenta il numero di minimi locali e quindi diminuisce la probabilità p_1 di trovare una soluzione ottima;

v) dalla mancanza di una caratterizzazione del minimo globale semplice da verificare: questo non consente una facile definizione dei *criteri di stop* per i metodi iterativi di minimizzazione. (Un criterio di stop è una condizione che, se verificata, permette di arrestare l'esecuzione dell'algoritmo). La costruzione di un appropriato criterio di stop per un algoritmo di ottimizzazione globale risulta fondamentale in quanto un numero eccessivamente grande di passi può portare a costi computazionali notevoli, mentre un numero troppo piccolo di passi può non permettere di individuare un punto di minimo globale, quindi fornire un risultato non accettabile.

I metodi introdotti per risolvere il problema 1.1.1 sono numerosi e molto differenti tra loro. Esistono comunque caratteristiche comuni a tutti i metodi.

[1]MATLAB è un *trademark* di The Mathworks, Inc., e-mail: info@mathworks.com, www: http://www.mathworks.com

2.2.1 Convergenza

Un qualunque metodo di ottimizzazione globale deve garantire la convergenza della successione dei punti generati alla soluzione esatta del problema.

A tal proposito, l'algoritmo di minimizzazione sia definito da una famiglia di funzioni A_i, ossia:

$$A = (A_1, A_2, \ldots);$$

il punto corrente utilizzato dalla procedura è dato da:

$$x_i = A_i((x_j, y_j,\ j = 1, \ldots, i-1), I(P)),$$

dove $y_j = f(x_j)$ e $I(P)$ è l'informazione disponibile per una classe P di problemi, aggiuntiva all'essere f continua; ad esempio $I(P)$ potrebbe essere la conoscenza della costante di Lipschitz.

Normalmente, dopo k osservazioni viene assunto come minimo globale il più piccolo tra i valori osservati, cioè:

$$y_k^* = \min_{1 \le i \le k} y_i.$$

Definizione 2.2.1 (Convergenza di un algoritmo). *Un algoritmo di minimizzazione globale A è convergente se*

$$y_k^* \underset{k \to \infty}{\longrightarrow} \min_{x \in S} f(x)$$

Una condizione necessaria e sufficiente per la convergenza di un algoritmo di ottimizzazione globale è data dal seguente teorema (cfr. [71]).

Teorema 2.2.1. *Sia A un algoritmo per la minimizzazione di una funzione f continua su un insieme compatto S di $\mathbb{R}^n$. Allora A converge al minimo globale di f se e solo se la successione x_k dei punti di prova è* ovunque densa *in S.*

Dimostrazione. Se la funzione f è continua e la successione x_k è ovunque densa in S, la condizione sufficiente è ovvia. Per quanto riguarda la condizione necessaria, si supponga che l'algoritmo A converga al minimo globale di una funzione f continua ma che la successione x_k non sia ovunque densa in S. Allora esiste $\varepsilon > 0$ ed un punto $\bar{x} \in S$ tale che $\|\bar{x} - x_k\| > \varepsilon, \quad \forall k = 1, 2, \ldots$. Si consideri il cono

$$g(x) = \min_{x \in S} f(x) - \delta + \frac{\|\bar{x} - x\|}{\varepsilon} \left(\max_{x \in S} f(x) - \min_{x \in S} f(x) + \delta \right),$$

dove $\delta > 0$ e sia $F(x)$ la funzione così definita:

$$F(x) = (\min(f(x), g(x))).$$

Si osservi che $F(x)$ coincide con $f(x)$ ovunque in S tranne che nei punti interni alla sfera $\|x - \bar{x}\| \le \varepsilon$. Allora la successione dei punti di prova per la funzione obiettivo

$f(x)$ e per la funzione $F(x) = (\min(f(x), g(x)))$ è esattamente la stessa: infatti, per come si sono scelti $\bar{x}$ e ε, x_k dipende solo da x_i e da $f(x_i)$, $i = 1, \ldots, k-1$. Ma la successione

$$Y_k^* = \min_{1 \le i \le k} F(x_i)$$

non converge al minimo globale $F(\bar{x})$ di $F(x)$, poiché $F(x_i) - F(\bar{x}) \ge \delta, i = 1, 2, \ldots$; (infatti:

$$F(x_i) = f(x_i),$$

$$F(\bar{x}) = \min_{x \in S} f(x) - \delta$$

e

$$F(x_i) - F(\bar{x}) = f(x_i) - \min_{x \in S} f(x) + \delta \ge \delta).$$

La condizione necessaria è così dimostrata. □

2.2.2 Soluzioni approssimate

Un ulteriore elemento di difficoltà da considerare nell'affrontare il problema 1.1.1 è il fatto che, in generale:

il problema dell'ottimizzazione globale non è risolvibile con certezza assoluta in un numero finito di passi.

Questo risultato, formulato da Dixon nel 1978, si può dimostrare come segue (cfr. [14]).
Data una funzione f continuamente differenziabile e un intorno I di un punto $\tilde{x} \in S$, esiste una funzione f' tale che $f + f'$ sia un'*indentazione* di f, ossia:

- $f + f'$ sia ancora continuamente differenziabile;
- $f + f'$ sia uguale ad f su $S \setminus I$;
- $f + f'$ raggiunga il suo minimo globale, inferiore a quello di f, in $\tilde{x}$.

Allora, comunque si scelga $\tilde{x}$, non è possibile verificare con certezza se $\tilde{x}$ è un punto di minimo globale senza valutare la funzione in almeno un punto di ogni intorno I di $\tilde{x}$. Poiché I può essere scelto arbitrariamente piccolo, ne segue che un metodo di risoluzione potrebbe richiedere un numero infinito di passi per individuare il minimo globale di $f + f'$.

Questo significa che in generale non è possibile individuare un punto di S^* in un tempo finito. Il problema 1.1.1 si considera quindi risolto quando viene individuata un'approssimazione della soluzione esatta a meno di un'accuratezza positiva prefissata ε.

Generalmente, gli algoritmi di risoluzione individuano una soluzione del problema 1.1.1 come limite di una successione di soluzioni di problemi che approssimano il problema iniziale. L'introduzione di criteri di stop appropriati consente di ottenere l'approssimazione desiderata.

Riportiamo di seguito alcune delle definizioni di soluzione approssimata usate più frequentemente nell'ottimizzazione sia vincolata che non vincolata (cfr. [28]).

Definizione 2.2.2. *Sia x^* un punto di minimo locale o globale di f. Un punto $x \in S$ rappresenta una soluzione approssimata di x^* se*

$$\|x - x^*\| \leq \varepsilon.$$

Uno svantaggio nell'utilizzo dell'approssimazione secondo la definizione 2.2.2 consiste nel fatto che una piccola perturbazione sui dati può avere un effetto maggiore sulla localizzazione del minimo globale x^* piuttosto che sul suo valore. Si considera pertanto la seguente:

Definizione 2.2.3. *Sia x^* un punto di minimo locale o globale di f. Un punto $x \in S$ rappresenta una soluzione approssimata di x^* se*

$$|f(x) - f(x^*)| \leq \varepsilon.$$

Se si vuole approssimare un punto stazionario si può usare la seguente definizione.

Definizione 2.2.4. *Un punto x interno ad S rappresenta un'approssimazione di un punto stazionario di f se*

$$\|\nabla f(x)\| \leq \varepsilon.$$

Dalle condizioni di ottimalità di Kuhn-Tucker discende la seguente:

Definizione 2.2.5. *Sia $S = I^n = [0,1]^n$ il cubo unitario. Un punto $x \in S$, $x \equiv (x_i)$, rappresenta un'approssimazione di un punto di Kuhn-Tucker di f se:*

i) per ogni i tale che $x_i > 0$, $\partial f(x)/\partial x_i \leq \varepsilon$,

ii) per ogni i tale che $x_i < 1$, $\partial f(x)/\partial x_i \geq -\varepsilon$,

essendo $\partial f(x)/\partial x_i$ la derivata di f rispetto all'i-esima variabile calcolata in x.

Possibili alternative di soluzione approssimata sono date dalle seguenti:

Definizione 2.2.6. *Sia $S = I^n = [0,1]^n$ il cubo unitario. Un punto $x \in S$ rappresenta un'approssimazione di un punto di minimo locale di f se esiste un insieme aperto N contenente x tale che*

$$f(x) \leq f(z) + \varepsilon\|z - x\|,$$

$\forall z \in N \cap S$.

Definizione 2.2.7. *Sia $\varepsilon \in [0,1]$ e*

$$y^* = \max_{x \in S} f(x), \quad y_* = \min_{x \in S} f(x).$$

Un punto $x \in S$ rappresenta un'approssimazione di un punto di minimo locale di f se

$$|f(x) - y_*| \leq \varepsilon |y^* - y_*|.$$

Definizione 2.2.8. *Sia* $S = \{x \in \mathbb{R}^n \mid a_j x \leq d_j,\ j = 1, \ldots, m\}$ *con* x *vettore colonna di* $\mathbb{R}^n$, a_j *vettore riga di* $\mathbb{R}^n$ *e* $d_j \in \mathbb{R}$. *Sia*

$$I_\varepsilon(x) = \{j \in \{1, \ldots, m\} \mid a_j x - d_j + \varepsilon \geq 0\}.$$

Un punto $x \in S$ *rappresenta un'approssimazione di un punto di Kuhn-Tucker di* f *se* $I_\varepsilon(x)$ *è vuoto e*

$$\|\nabla f(x)\| \leq \varepsilon$$

oppure

$$y_\varepsilon(x) = \left(\left(A_{I_\varepsilon(x)} A^T_{I_\varepsilon(x)}\right)^{-1} A_{I_\varepsilon(x)}\right) \nabla f(x) \leq \varepsilon,$$

dove $A_{I_\varepsilon(x)} = (a_j)_{j \in I_\varepsilon(x)}$.

Definizione 2.2.9. *Sia* $x^* \in S$ *un punto di minimo globale o locale di* f. *Un punto* $x \in \mathbb{R}^n$ *rappresenta una soluzione approssimata di* x^* *se*

$$f_i(x) \leq \varepsilon,\ i = 1, \ldots, m,$$

$$f_0(x) - f^* \leq \varepsilon,$$

essendo f_i *una successione di punti costruita dall'algoritmo.*

I punti della successione presa in considerazione nella definizione 2.2.9 non sono necessariamente punti dell'insieme ammissibile S. La definizione suddetta garantisce che il punto limite della successione sia sufficientemente vicino ad S.

Le definizioni 2.2.3 e 2.2.4 possono essere messe in relazione tra loro dal lemma seguente (cfr. [28]).

Lemma 2.2.1. *Sia* $r > 0$ *e sia* $f : \mathbb{R}^n \to \mathbb{R}$ *una funzione due volte continuamente differenziabile. Inoltre la matrice hessiana di* f, *denotata con* $H(x)$, *sia semidefinita positiva* $\forall x \in B(x^*, r)$, *essendo* $\nabla f(x^*) = 0$. *Se* $x \in B(x^*, r)$ *approssima* x^* *secondo la definizione 2.2.4 con* $\|\nabla f(x^*)\| \leq \varepsilon' = \varepsilon / r$, *allora* x *approssima* x^* *secondo la definizione 2.2.3.*

In [74], Vavasis ha dimostrato il seguente risultato per le definizioni 2.2.5 e 2.2.6:

Lemma 2.2.2. *Sia* $S = I^n = [0, 1]^n$ *ed* $f \in C^1[0, 1]^n$. *Se* x *è un'approssimazione di un minimo locale di* f *secondo la definizione 2.2.6 allora lo è anche secondo la definizione 2.2.5. Viceversa, se* x *soddisfa la definizione 2.2.5 allora soddisfa anche la definizione 2.2.6,* $\forall \varepsilon' > \varepsilon$.

Queste numerose definizioni di soluzione approssimata si sono rese necessarie per la grande varietà di algoritmi proposti per la risoluzione del problema 1.1.1; l'idea comune è quella di scegliere il criterio di stop più adatto a garantire la convergenza dell'algoritmo.

2.2.3 Classificazione dei metodi di ottimizzazione

Tra tutte le classificazioni presenti in letteratura, consideriamo quella proposta da Dixon e Szegö (cfr. [16], [15]). Secondo questa classificazione, i metodi sviluppati per risolvere il problema 1.1.1 si possono suddividere in linea di massima in due classi:

1. metodi deterministici;
2. metodi stocastici.

I metodi deterministici forniscono un'assoluta garanzia di successo utilizzando le informazioni sulla funzione obiettivo ottenute durante l'esecuzione dell'algoritmo, ma inevitabilmente richiedono ipotesi molto restrittive sulla f: ad esempio, spesso si può richiedere che la funzione sia lipschitziana in S.

I metodi stocastici sono quei metodi in cui il processo di minimizzazione dipende in parte da eventi probabilistici, nel senso che i risultati forniti sono essi stessi variabili aleatorie. In molti casi la funzione viene valutata su un insieme di punti scelti a caso in S; questa valutazione può comportare un successivo lavoro di manipolazione del campione.

Una conseguenza dell'introduzione di elementi di tipo probabilistico in un algoritmo di tipo deterministico è il sacrificio dell'assoluta garanzia di successo del metodo.

Ovviamente tutti i metodi, stocastici o deterministici che siano, non possono prescindere dall'assicurare una qualche forma di convergenza. In alcuni casi si può richiedere che un metodo sia competitivo dal punto di vista empirico pur senza fornire garanzie di convergenza (*metodo euristico*), ma questo non è affatto soddisfacente dal punto di vista numerico. Idealmente si vorrebbe assicurare la possibilità di individuare un'approssimazione del minimo globale in un numero finito di passi e, sotto determinate ipotesi, alcuni metodi deterministici sono in grado di fornire una simile garanzia. Nella maggioranza dei casi però il massimo che si può pretendere è una convergenza di tipo *asintotico*, ossia con uno sforzo computazionale tendente all'infinito. In generale la convergenza dei metodi stocastici è proprio di questo tipo; in particolare essi garantiscono una *convergenza in probabilità* o *con probabilità 1*.

Definizione 2.2.10 (Convergenza in probabilità). *Una successione* $\{x_k\}$ *converge in probabilità verso* x *(*$x_k \xrightarrow{p} x$*) se,* $\forall \varepsilon > 0$*,*

$$\lim_{k\to\infty} p(|x_k - x| \leq \varepsilon) = 1.$$

Definizione 2.2.11 (Convergenza con probabilità 1). *Una successione* $\{x_k\}$ *converge con probabilità 1, o quasi certamente, a* x *(*$x_k \xrightarrow{q.c.} x$*) se:*

$$p\left(\lim_{k\to\infty} x_k = x\right) = 1.$$

Quest'ultima definizione non richiede che la convergenza avvenga per tutti i punti ma solamente per un insieme a cui è assegnata probabilità unitaria. Nella successiva trattazione e quando non diversamente precisato, per convergenza asintotica si intenderà la convergenza in probabilità.

Sotto ipotesi molto deboli sulla funzione e sulla distribuzione di probabilità usata per il campionamento, è possibile dimostrare che la probabilità che un metodo stocastico determini un'approssimazione dell'ottimo globale si avvicina ad uno al crescere della numerosità campionaria (Solis e Wets, 1981, [65]).

Tipicamente, alla classe dei metodi deterministici appartengono i metodi del tipo *Branch and Bound*, come ad esempio il *metodo Grid Search* (cfr. [4]), gli algoritmi di Kushner (1964, [38]), di Pijavskii (1972, [51]), di Zilinskas (1981, [76]) e di Strongin (1984, [67]), i metodi di partizione su intervalli introdotti da Moore (1966, [48]) (si veda [31], [53]), i metodi basati sulle curve di Peano (Strongin, 1984, [67]; Sergeyev e Grishagin, 1994, [64], Sergeyev e Strongin, 2000, [68]) e le strategie di partizione per funzioni lipschitziane (cfr. [52], [32]). Alla stessa classe appartengono anche alcuni metodi del tipo *Improvement of Local Minima*, per cui si rimanda al paragrafo 3.7. Per ulteriori riferimenti bibliografici in proposito si vedano [52], [72], [34], [68].

In generale, un metodo si caratterizza come stocastico se:

1) i punti di prova vengono generati secondo una particolare distribuzione di probabilità su S (metodi *Two-Phase*, *Random Search* e *Simulated Annealing*) o lungo direzioni scelte a caso in $\mathbb{R}^n$ (metodi *Random Directions*); oppure:

2) la funzione obiettivo f viene considerata come la realizzazione di un *processo stocastico* definito a priori (metodi *Random Function*).

Per entrambe le classi 1) e 2) esistono numerosi algoritmi efficienti (cfr. [30], [62], [68]). Una breve descrizione dei metodi stocastici più interessanti per la trattazione successiva verrà data al capitolo seguente.

Ovviamente non è possibile affermare che i metodi stocastici siano assolutamente superiori a quelli deterministici; è però vero che per problemi di grande dimensione l'uso di tecniche stocastiche è spesso l'unico approccio possibile.

2.3 Studio dell'informazione associata al problema

Un nuovo tipo di approccio, volto a superare le difficoltà precedentemente elencate nella risoluzione del problema dell'ottimizzazione globale, mira ad analizzare l'*informazione* associata al problema (cfr. [28]). L'utilità di questo approccio consiste nell'individuare le informazioni utilizzate dall'algoritmo durante la sua esecuzione, in modo da stabilire delle limitazioni entro le quali si ritiene che tale algoritmo sia efficiente.

Chiaramente, ci si aspetta che a un maggior numero di informazioni corrisponda un algoritmo avente migliori proprietà di convergenza.
Nel caso del problema 1.1.1, ci si può basare ad esempio su informazioni "a priori" fornite dalla definizione stessa del problema, come la differenziabilità con continuità della funzione obiettivo, oppure sulla disponibilità di informazioni aggiuntive, come il valore del minimo globale, il numero dei minimi locali, il valore della costante di Lipschitz, la conoscenza di qualche limitazione sulle derivate prime o seconde o sulla misura delle regioni di attrazione dei minimi locali. Tali informazioni costituiscono informazioni *globali* sulla funzione obiettivo. Raramente tali informazioni sono disponibili; ne consegue una limitata applicabilità dei metodi che ne fanno uso.

Per informazioni *locali* si intendono quelle informazioni prodotte nel corso dell'esecuzione dell'algoritmo, come ad esempio i valori della funzione nei punti generati. I metodi che sfruttano informazioni locali si applicano a una classe più ampia di problemi; anche in questo caso però è indispensabile conoscere delle informazioni globali *di base* per poter applicare l'algoritmo, nonché produrre informazioni globali sulla funzione per garantire una qualche forma di convergenza.

Quando la funzione obiettivo è continua, l'informazione a priori *di base* da conoscere è la procedura che fornisce il valore della funzione in un punto dato e l'insieme ammissibile S in cui il minimo globale va ricercato. Nella maggior parte dei casi sono disponibili poche altre informazioni a priori.

Nel caso in cui solo le informazioni di base siano note, il risultato del metodo sarà completamente determinato dai punti di prova scelti in S e dai loro valori. Se il numero dei punti di prova è fissato, i metodi di ottimizzazione potrebbero differire tra loro solo per la distribuzione di tali punti in S.

La particolare scelta dei punti in S è finalizzata al raggiungimento di due obiettivi concorrenti.

Da un lato si vuole poter realizzare una ricerca globale su S, in quanto il minimo globale può trovarsi in un punto qualunque dell'insieme ammissibile, quindi nessuna regione può essere trascurata a priori. Effettuare un'analisi di tipo globale significa ad esempio scegliere i punti in maniera uniforme su S in modo da ricoprire per intero l'insieme ammissibile.

D'altra parte è necessario effettuare anche un'analisi di tipo locale, dato che la probabilità di trovare un nuovo punto con valore della funzione inferiore all'ottimo corrente è maggiore in un intorno di un punto con valore relativamente basso piuttosto che con valore relativamente alto. Il procedimento ideale da seguire in questo caso è quello di scegliere i punti maggiormente concentrati in un intorno del miglior punto noto.

Le due strategie descritte prendono il nome rispettivamente di *strategia globale* e *strategia locale*.

Sia la ricerca globale che quella locale sono quindi di fondamentale importanza nella progettazione di un metodo di ottimizzazione efficiente: se per un verso non è possibile prescindere da un'analisi globale dell'insieme ammissibile in quanto i minimi locali possono trovarsi ovunque in S, dall'altro la ricerca locale è di grande

utilità sia quando i minimi si concentrano in una particolare regione che quando si vuole trovare un'approssimazione più accurata del valore dell'ottimo individuabile tramite la sola strategia globale.

In [66], Stephens e Baritompa analizzano l'informazione associata al problema 1.1.1 definendo il concetto di *informazione locale* per un'opportuna famiglia di funzioni.
Preliminarmente, si consideri la seguente definizione:

Definizione 2.3.1 (Classe ricca). *Una classe di funzioni* $\mathbb{F}$ *non vuota è detta* sufficientemente ricca *se:*

- *le funzioni* $f \in \mathbb{F}$ *sono continue;*
- $\forall y \in \mathbb{R},\ x \in S,\ \forall f \in \mathbb{F},\ \forall N$, *con* N *insieme aperto di* S *contenente* x, $\exists g \in \mathbb{F}$ *tale che* $g(x) = y$ *e* $g_{|_{S \setminus N}} = f_{|_{S \setminus N}}$.

Un'informazione locale può essere definita nel modo seguente:

Definizione 2.3.2 (Informazione locale). *Sia* S_{finito} *l'insieme di tutte le successioni finite in* S. *L'*informazione locale *per una famiglia* $\mathbb{F}$ *è una funzione* LI *definita su* $\mathbb{F} \times S_{finito}$ *tale che:*

$$se \quad g_{|_{S \setminus N}} = f_{|_{S \setminus N}} \ allora \quad LI(f, X) = LI(g, X),$$

$\forall f, g \in \mathbb{F},\ \forall X \subset S_{finito},\ \forall N$ *insieme aperto di* S *contenente* X.

Utilizzando la nozione di informazione locale è possibile dare una definizione rigorosa per i concetti di *algoritmo deterministico* e *algoritmo stocastico.*

Definizione 2.3.3 (Algoritmo deterministico). *Sia* $\{x_0, x_1, x_2, \ldots\}$ *una successione di punti generata da un algoritmo e sia* $X_k = \{x_0, x_1, \ldots, x_k\}$ *una sua sottosuccessione finita. Un algoritmo* deterministico a campionamento sequenziale *su una classe di funzioni* $\mathbb{F}$, *è un algoritmo per il quale esiste una funzione informazione locale* LI *tale che, qualunque sia la* $f \in \mathbb{F}$ *alla quale l'algoritmo viene applicato,* x_{k+1} *dipende solo da* $LI(f, X_k)$.

In altri termini, un algoritmo deterministico individua, ad ogni passo, un punto che dipende esclusivamente dall'informazione accumulata fino a quel momento.
Il teorema seguente afferma che gli algoritmi deterministici forniscono un'assoluta garanzia di successo quando il campionamento è infinito.

Teorema 2.3.1. *Sia* A *un algoritmo deterministico a campionamento sequenziale su una classe di funzioni* $\mathbb{F}$ *sufficientemente ricca. Sia* $\overline{X}_f$ *la chiusura dell'insieme dei punti individuati da* A *quando applicato ad una funzione* $f \in \mathbb{F}$ *e sia* X_f^* *l'insieme dei punti di minimo globale di* f. *Allora* A *individua un punto di minimo globale* $\forall f \in \mathbb{F}$ *(cioè* $\overline{X}_f \cap X_f^* \neq \emptyset$*) se e solo se* $\overline{X}_f = S,\ \forall f \in \mathbb{F}$.

Definizione 2.3.4 (Algoritmo stocastico). *Sia* $\{x_0, x_1, x_2, \ldots\}$ *una successione di punti generata da un algoritmo e sia* $X_k = \{x_0, x_1, \ldots, x_k\}$ *una sua sottosuccessione finita. Un algoritmo* stocastico a campionamento sequenziale *su una classe di funzioni* $\mathbb{F}$*, è un algoritmo per il quale esiste una funzione informazione locale* LI *tale che, qualunque sia la* $f \in \mathbb{F}$ *alla quale l'algoritmo viene applicato,* x_{k+1} *dipende da* $LI(f, X_k)$ *e dal valore* ω_{k+1} *di una variabile aleatoria.*

In altri termini, il risultato di un algoritmo stocastico dipende, oltre che dall'informazione accumulata fino a quel momento, anche da fattori casuali.

Teorema 2.3.2. *Per ogni algoritmo stocastico a campionamento sequenziale esiste una funzione* $f \in \mathbb{F}$ *tale che la probabilità che l'algoritmo individui il minimo globale di* f *è inferiore a* ε*, con* ε *costante positiva.*

Il teorema 2.3.2 evidenzia come la difficoltà di risoluzione del problema dell'ottimizzazione globale non sia superata dall'introduzione di algoritmi stocastici, proprio perché in alcuni casi la probabilità di localizzare un punto di minimo globale può essere arbitrariamente piccola.

Ovviamente, la disponibilità di informazioni di tipo globale, di cui innanzitutto si dovrebbe dare una definizione analoga alla 2.3.2, sarebbe estremamente utile per comprendere a fondo le caratteristiche peculiari degli algoritmi.
D'altra parte però, persino conoscere informazioni globali non garantisce l'esistenza di algoritmi efficienti; si veda a tal proposito l'esempio 4.3.1.

Capitolo 3

Metodi stocastici

In questo capitolo verranno descritti alcuni metodi tipicamente stocastici, quali i metodi *Two-Phase*, *Random Search*, *Simulated Annealing* e *Random Directions*, nonché alcuni metodi del tipo *Improvement of Local Minima* che possono essere inquadrati in questa classe.

Il problema 1.1.1 verrà considerato risolto quando sarà trovata un'approssimazione del minimo globale secondo le definizioni 2.2.2 e 2.2.3, ossia un punto in:

$$B_\varepsilon(S^*) = \{x \in S : \|x - x^*\| \leq \varepsilon\}, \tag{3.1}$$

oppure nell'insieme:

$$B_\varepsilon(f) = \{x \in S : |f(x) - f^*| \leq \varepsilon\}, \tag{3.2}$$

dove $x^* \in S^*,\ f(x^*) = f^*$ e $\varepsilon > 0$.

3.1 Introduzione

Nel caso più generale, i passi fondamentali di un algoritmo stocastico sono i seguenti:

campionamento: un certo numero di punti viene generato nello spazio ammissibile S mediante una procedura stocastica e per ciascuno viene calcolato il corrispondente valore della f;

ottimizzazione: viene applicata una procedura di minimizzazione locale ad alcuni dei punti campionati al passo precedente, in modo da individuare i minimi locali delle regioni di attrazione a cui tali punti appartengono;

verifica di un criterio di stop: l'esecuzione dell'algoritmo viene interrotta nel momento in cui un criterio prestabilito risulta verificato.

In base alla diversa realizzazione di ciascun passo, è possibile distinguere diverse classi di metodi stocastici; si considerino ad esempio le seguenti possibili scelte per ogni passo (cfr. [62]):

- campionamento:

  i) cardinalità del campione:
  - il numero dei punti da generare è prefissato;
  - la cardinalità del campione è determinata in modo adattivo durante l'esecuzione dell'algoritmo;

  ii) strategia di campionamento:
  - i punti scelti in S, ad esempio mediante campionamento uniforme, sono indipendenti ed identicamente distribuiti;
  - il campionamento avviene secondo una distribuzione predefinita in un intorno del punto corrente, ad esempio una sfera o un ipercubo;
  - i nuovi punti vengono generati in base ad una distribuzione che dipende non solo dal punto corrente, ma anche dai punti precedentemente generati e/o dai valori corrispondenti della funzione obiettivo;
  - dopo aver scelto una direzione d in $\mathbb{R}^n$ secondo una distribuzione di probabilità prefissata, i nuovi punti vengono generati in S lungo d mediante una opportuna applicazione;

- ottimizzazione:
  - non si applicano procedure di minimizzazione;
  - si applica una procedura di minimizzazione a n dei punti generati;
  - si applica una procedura di minimizzazione a tutti i punti generati;

- scelta del criterio di stop:

  l'esecuzione dell'algoritmo termina quando:
  - l'ottimo globale, noto, è stato raggiunto a meno di una accuratezza prefissata;
  - viene raggiunto un numero massimo di iterazioni o di valutazioni di funzione;
  - nelle ultime iterazioni non si ottiene un sensibile miglioramento dell'ottimo corrente;
  - una stima *a posteriori* della probabilità che non esistano altri minimi locali, oltre a quelli già individuati, supera una soglia prefissata;

- una stima *a posteriori* della probabilità che nelle successive iterazioni si possano ottenere ulteriori miglioramenti dell'ottimo corrente (considerando sia la possibilità di individuare un nuovo minimo locale che lo sforzo da mettere in atto per raggiungere l'obiettivo) è inferiore ad una soglia prefissata.
- è stata superata una soglia imposta al tempo di calcolo.

3.2 Pure Random Search

Il più semplice tra tutti i metodi stocastici per l'ottimizzazione globale è il *Pure Random Search* (PRS) (cfr. [30]). L'algoritmo PRS consiste nel generare una successione di punti indipendenti e identicamente distribuiti con distribuzione uniforme su S. La valutazione della funzione obiettivo in ciascuno di questi punti consente di identificare il punto di minimo tra i punti campionati. Tale punto è usato come approssimazione della soluzione ottima. Lo schema dell'algoritmo è il seguente:

Algoritmo 3.2.1 (Pure Random Search, PRS).

sia $k \leftarrow 0$ e si generi y con un campionamento uniforme su S;
si ponga $x_0 \leftarrow y$ e $fx \leftarrow f(y)$;
loop
  si scelga y con un campionamento uniforme su S;
  if $f(y) < fx$
    $x_{k+1} \leftarrow y$; $fx \leftarrow f(y)$;
  else
    $x_{k+1} \leftarrow x_k$;
  endif
  $k \leftarrow k + 1$;
forever
end

Per quanto riguarda le proprietà di convergenza, consideriamo la probabilità che in n iterazioni l'algoritmo individui un punto in $B_\varepsilon(S^*)$:

$$\begin{aligned} p(x_i \in B_\varepsilon(S^*) \text{ per un } 1 \leq i \leq n) &= 1 - p(x_1, \ldots, x_n \notin B_\varepsilon(S^*)) \\ &= 1 - (1 - \varphi(B_\varepsilon(S^*)))^n, \end{aligned} \tag{3.3}$$

dove φ rappresenta la distribuzione uniforme su S. Avendo ipotizzato f continua su S compatto, $\varphi(B_\varepsilon(S^*)) > 0$, quindi la (3.3) tende ad 1 al tendere di n ad infinito. In altri termini, l'algoritmo offre un successo garantito in probabilità.
Inoltre si può dimostrare il seguente risultato (cfr. [30]):

Proposizione 3.2.1 (Devroye, 1978). *La successione $f(x_k)$ converge al minimo globale con probabilità 1 al crescere di k, cioè :*

$$p\left\{\lim_{k\to\infty} f(x_k) = f^*\right\} = 1.$$

Nonostante lo schema molto semplice, l'utilizzo del PRS è molto limitato: infatti né i valori ottenuti per f né le eventuali informazioni sulla struttura della funzione vengono sfruttati, quindi l'informazione prodotta dall'algoritmo viene persa. Ne consegue che il PRS risulta poco efficiente: il numero di punti richiesto prima che un punto generato possa essere considerato un'approssimazione del minimo globale cresce in modo esponenziale con la dimensione del problema (si vedano i risultati sulla complessità al paragrafo 4.3.2).

Lo sforzo computazionale necessario per ottenere un *improvement* dell'ottimo corrente mediante una ricerca casuale può essere ridotto quando la funzione obiettivo è abbastanza "liscia": in questo caso l'uso di procedure di minimizzazione risulta più efficiente. Un primo tentativo in questo senso è rappresentato dal metodo *Single Start*: questo metodo differisce dal PRS per il fatto che una singola procedura di ricerca locale viene applicata al punto con valore più basso generato dall'algoritmo (cfr. [62]).

Algoritmo 3.2.2 (Single Start).

sia $k \leftarrow 0$ e si generi y con un campionamento uniforme su S;
si ponga $x_0 \leftarrow y$ e $fx \leftarrow f(y)$;
loop
  si scelga y con un campionamento uniforme su S;
  if $f(y) < fx$
    $x_{k+1} \leftarrow y$; $fx \leftarrow f(y)$;
  else
    $x_{k+1} \leftarrow x_k$;
  endif
  $k \leftarrow k + 1$;
forever
si applichi a x_k una procedura di ricerca locale e si determini x^*;
si calcoli $f(x^*)$;
end

A partire dal PRS e dal Single Start, sono stati proposti i cosiddetti *metodi Two-Phase*. Per questi metodi in genere si ipotizza che esista solo un numero finito di minimi locali, tutti interni a S, e che la funzione obiettivo f sia due volte differenziabile con derivate continue. Esempi di metodi Two-Phase sono il metodo *Multistart*, i *metodi di clustering* e i *metodi Multi Level*.

3.3 Metodi Two-Phase

I *metodi Two-Phase*, introdotti da Rinnooy Kan e Timmer (cfr. [54], [55]), vengono così chiamati in quanto la ricerca di un minimo globale si articola essenzialmente in due fasi:

i) (*global phase*): viene generato un campione di punti con distribuzione uniforme in S;

ii) (*local phase*): per ognuno di questi punti viene determinato il minimo locale della regione di attrazione alla quale il punto appartiene e ogni minimo locale trovato è considerato un candidato ad essere minimo globale. La determinazione del minimo locale è eseguita usando una procedura di ricerca locale.

3.3.1 Multistart

Il *Multistart* è il più semplice tra i metodi Two-Phase: in questo caso la procedura di ricerca locale è applicata a ciascuno dei punti del campione generato. Lo schema dell'algoritmo è il seguente:

Algoritmo 3.3.1 (Multistart).

sia $k \leftarrow 0$ e si generi y con un campionamento uniforme su S;
si applichi a y una procedura di ricerca locale e si determini y';
si ponga $x_0 \leftarrow y'$ e $fx \leftarrow f(y')$;
loop
  si generi un punto y secondo la distribuzione uniforme su S;
  si applichi a y una procedura di ricerca locale e si determini y';
  if $f(y') < fx$
    $x_{k+1} \leftarrow y'$; $fx \leftarrow f(y')$;
  else
    $x_{k+1} \leftarrow x_k$;
  endif
  $k \leftarrow k + 1$;
forever
end

Questo metodo, come il PRS, risulta affidabile in quanto assicura la convergenza al minimo globale con probabilità uno, però è ancora molto carente per quanto riguarda l'efficienza. La ragione principale è che, inevitabilmente, si troveranno gli stessi minimi locali più volte. Essendo le procedure di ricerca locale la parte più costosa in termini di tempo, tali procedure andrebbero utilizzate non più di una volta per ogni regione di attrazione di un minimo locale. I *metodi di clustering* e i *metodi Multi Level* sono stati introdotti nell'intento di migliorare l'efficienza del Multistart.

3.3.2 Metodi di clustering

L'idea di base dei metodi di clustering consiste nel partire da un campione uniforme su S per poi creare i *clusters*, cioè gruppi di punti mutuamente vicini, e applicare la procedura di ricerca locale non più di una volta per ciascuno di questi gruppi. Si vorrebbe idealmente che ciascun cluster corrispondesse ad una regione di attrazione di un minimo locale della funzione obiettivo. Sono state proposte due tecniche per creare i clusters dal campione iniziale. La prima, introdotta da Becker e Lago in [6], è detta *riduzione* in quanto conserva solo una frazione γ del campione, frazione che comprende i punti per cui è più basso il valore della funzione obiettivo. La seconda, studiata da Törn in [70], è detta *concentrazione* e trasforma il campione effettuando una o più *procedure di più ripida discesa* a partire da ogni punto. Molto spesso la tecnica di riduzione del campione viene preferita a quella di concentrazione, in quanto i punti sono distribuiti uniformemente su S. In entrambi i casi, i clusters sono formati per passi, partendo da un *punto-seme*, che può essere:

i) il punto con il valore più basso della funzione non appartenente a un cluster;

ii) il minimo locale trovato applicando una procedura di ricerca locale al punto col valore più basso della funzione.

I punti vengono aggiunti al cluster applicando una particolare procedura di *clustering*.

Uno dei più famosi metodi di clustering è il *Single Linkage Clustering* (Rinnooy Kan e Timmer, [54]). In questo metodo i clusters vengono formati in modo sequenziale. Ogni cluster viene inizializzato con un punto-seme. Una volta che il cluster C è stato inizializzato, si cerca un punto x, non appartenente al cluster, tale che la sua distanza d dal cluster C soddisfi la seguente:

$$d(x, C) = \min_{y \in C} \|x - y\|.$$

Questo punto viene aggiunto al cluster e il procedimento viene ripetuto fintanto che $d(x, C)$ resta inferiore ad una *distanza critica* r_k così definita:

$$r_k = \frac{1}{\sqrt{\pi}} \left(\Gamma\left(1 + \frac{n}{2}\right) \cdot m(S) \cdot \frac{\xi \ln(kN)}{kN} \right)^{\frac{1}{n}}, \tag{3.4}$$

dove N è la numerosità del campione, $\xi > 0$ e Γ è la *funzione gamma*.
Lo schema dell'algoritmo è il seguente:

Algoritmo 3.3.2 (Single Linkage Clustering).

siano $k \leftarrow 1$ e $X^* = \emptyset$ l'insieme dei punti-seme;
loop
si generino N punti $x_{(k-1)N+1}, \ldots, x_{kN}$ secondo la distribuzione uniforme su S;

si determini il *campione ridotto* S_k contenente i γkN punti migliori del campione $x_1, \ldots, x_{kN}$;
$j \leftarrow 1$;
repeat
if $j > |X^*|$
si selezioni come successivo punto-seme il migliore punto $\bar{x} \in S_k$ non ancora incluso in un cluster;
si applichi ad $\bar{x}$ una procedura di ricerca locale L per trovare un minimo locale x^*;
si aggiunga x^* a X^*;
else if $j \leq |X^*|$
si selezioni il j-esimo minimo locale in X^* come successivo punto-seme;
si aggiungano al cluster appena inizializzato tutti i punti di S_k non ancora clusterizzati e con distanza inferiore a r_k da un qualunque punto del cluster;
$j \leftarrow j + 1$;
endif
until $\nexists\, x \in S_k$ non assegnato ad un cluster
$k \leftarrow k + 1$;
forever
end

Si noti che la scelta di r_k in (3.4) garantisce che, se $\xi > 2$, la probabilità che il Single Linkage utilizzi una procedura L all'iterazione k tende a zero al crescere di k. Inoltre si può dimostrare che, se $\xi > 4$, anche se il campionamento continua all'infinito, il numero totale delle ricerche locali eseguite è finito con probabilità uno (cfr. [54]).

Queste proprietà assicurano l'efficienza del metodo in termini di numero di ricerche locali eseguite; in tale procedimento si perde però la garanzia asintotica di successo. Infatti il Single Linkage riesce a trovare con certezza un minimo locale in ogni componente connessa dell'insieme di livello:

$$L(f, y_\gamma) = \{x \in S : f(x) \leq y_\gamma\}, \tag{3.5}$$

dove y_γ è scelto in modo tale che γ sia la misura dell'insieme di livello indotta dalla distribuzione uniforme; ciascuna di queste componenti potrebbe però contenere più di una regione di attrazione, per cui alcuni minimi locali potrebbero non essere trovati. I metodi *Multi Level* sono stati introdotti proprio per ripristinare l'affidabilità nei metodi di clustering.

3.3.3 Metodi Multi Level

Tra i metodi Multi Level, il *Multi Level Single Linkage* (MLSL) è un metodo che combina l'efficienza delle tecniche di clustering con le proprietà teoriche di affida-

bilità del Multistart. In questo caso la procedura di ricerca locale è applicata ad ogni punto del campione, ad eccezione del caso in cui ci sia un altro punto del campione a distanza inferiore rispetto a una distanza critica r_k fissata, per il quale la funzione obiettivo abbia un valore più piccolo.
Lo schema dell'algoritmo è il seguente:

Algoritmo 3.3.3 (Multi Level Single Linkage).

siano $k \leftarrow 1$ e $X^* = \emptyset$;
loop
si generino N punti $x_{(k-1)N+1}, \ldots, x_{kN}$ secondo la distribuzione uniforme su S;
$i \leftarrow 1$;
repeat
if $\nexists\ j = 1, \ldots, kN$ tale che $f(x_j) < f(x_i)$ e $\|x_j - x_i\| < r_k$
si applichi a x_i una procedura di ricerca locale L e si aggiunga a X^* il minimo locale trovato;
endif
$i \leftarrow i + 1$;
until $i > kN$
$k \leftarrow k + 1$;
forever
end

La distanza critica r_k è ancora quella definita dalla (3.4).

Nonostante la sua semplicità, le proprietà teoriche dell'algoritmo 3.3.3 sono piuttosto forti: Rinnooy Kan e Timmer in [55] hanno dimostrato che:

i) se x è un punto arbitrario del campione, la probabilità di applicare la procedura di ricerca locale L a questo punto, all'iterazione k, tende a zero al crescere di k;

ii) se $\xi > 2$, la probabilità che venga eseguita una ricerca locale all'iterazione k, tende a zero al crescere di k;

iii) se $\xi > 4$, il numero totale di ricerche locali eseguite dal MLSL è finito con probabilità 1;

iv) un minimo locale viene trovato dal MLSL in un numero finito di iterazioni con probabilità uno.

In conclusione, si noti che i metodi Two-Phase mirano a trovare tutti i minimi locali del problema 1.1.1. Questa strategia non è la migliore, specie se il numero dei minimi locali è molto grande. I metodi *Random Search* vengono introdotti per trovare direttamente il minimo globale.

3.4 Metodi Random Search

La classe dei metodi Random Search, introdotta da Solis e Wets in [65], è costituita da algoritmi che generano successioni di numeri in S secondo una distribuzione di probabilità assegnata o successioni di distribuzioni di probabilità. Negli algoritmi più semplici i punti vengono generati a partire da un'unica distribuzione di probabilità, cioè tali punti sono variabili aleatorie indipendenti e identicamente distribuite. Alternativamente questa distribuzione può essere aggiornata in modo *adattivo*, ossia modificata nel corso dell'esecuzione dell'algoritmo con adattamenti che dipendono dall'iterazione e dai punti trovati alle precedenti iterazioni; in tal caso gli algoritmi si dicono di tipo *Random Search adattivi*.

La caratteristica più importante di questi metodi è il fatto che ad ogni iterazione corrisponde un miglioramento forzato del valore della funzione obiettivo.

Sia $\{\mu_k\}_{k=0}^{\infty}$ una successione di distribuzioni di probabilità in $\mathbb{R}^n$. Si consideri il seguente algoritmo:

Algoritmo 3.4.1 (Random Search).

siano $k \leftarrow 0$ e $x_0 \in S$;
loop
  si generi y_{k+1} secondo la distribuzione μ_k;
  $x_{k+1} \leftarrow D(x_k, y_{k+1})$;
  $k \leftarrow k + 1$;
forever
end

La funzione $D : S \times \mathbb{R}^n \to S$ soddisfa le seguenti condizioni:

$$f(D(x, y)) \leq f(x), \tag{3.6}$$

e, se $y \in S$,

$$f(D(x, y)) \leq f(y). \tag{3.7}$$

Queste condizioni assicurano che la successione $\{f(x_k)\}_{k=0}^{\infty}$ sia monotona decrescente con probabilità uno, dunque convergente a f^* per $k \to \infty$ (Solis e Wets, 1981).

Gli algoritmi di tipo Random Search come il 3.4.1 sono di natura *concettuale*, nel senso che attualmente non ne esiste ancora un'implementazione veramente efficiente. Comunque i risultati teorici che si possono ottenere per questi algoritmi sono interessanti di per se stessi e sono in grado di suggerire algoritmi effettivamente implementabili per l'ottimizzazione globale (Guus, Boender e Romeijn, [30]).

Anche il Pure Random Search, descritto nel paragrafo precedente, può essere inserito nella classe dei metodi Random Search; è interessante notare che il risultato di convergenza visto per il PRS continua a valere se si sostituisce la distribuzione uniforme con un'altra distribuzione, il cui supporto abbia intersezione non vuota con l'insieme S^* (Guus, Boender e Romeijn, [30]).

Si considerino ora due classi di algoritmi Random Search adattivi: il *Pure Adaptive Search* e l'*Adaptive Search*.

3.4.1 Pure Adaptive Search

Secondo lo schema del Pure Adaptive Search (PAS), a ciascuna iterazione viene generato un punto secondo una distribuzione uniforme in un sottoinsieme di punti "migliori" rispetto all'iterazione precedente. Ossia: alla generica iterazione k, il nuovo punto x_{k+1} della successione viene scelto uniformemente in un sottoinsieme di S contenente solo punti per i quali la funzione obiettivo ha valore strettamente minore del valore assunto in x_k. Tale sottoinsieme S_k di S è detto *improving region* ed è definito come segue:

$$S_k = \{x \in S : f(x) < f(x_{k-1})\}. \tag{3.8}$$

Più formalmente, lo schema dell'algoritmo è il seguente:

Algoritmo 3.4.2 (Pure Adaptive Search, PAS).

siano $k \leftarrow 0$ e $S_0 \leftarrow S$;
loop
  si generi x_{k+1} con un campionamento uniforme in S_k;
  $\omega_{k+1} \leftarrow f(x_{k+1})$;
  sia $S_{k+1} = \{x \in S : \ f(x) \leq \omega_{k+1}\}$;
  $k \leftarrow k + 1$;
forever
end

È possibile dimostrare che il numero atteso di iterazioni per il PAS cresce al più linearmente con la dimensione del problema (Zabinsky e Smith, [77], cfr. paragrafo 4.3.3). Nonostante questo risultato consenta di ipotizzare l'esistenza di un metodo PAS efficiente per l'ottimizzazione globale, sfortunatamente in pratica si presentano grosse difficoltà nella sua effettiva implementazione, in particolare nella costruzione della regione S_k e quindi nella generazione di un punto con distribuzione uniforme in S_k.

Una possibile soluzione per superare queste difficoltà potrebbe essere quella di introdurre un criterio *di accettazione-rifiuto*, che consiste nel generare punti uniformemente in S finché non si trovi un punto in S_k. In questo modo però si ritorna al PRS, quindi a un algoritmo non efficiente. La soluzione più promettente sembrerebbe essere quella di generare i punti in S in modo differente: anziché considerare una distribuzione uniforme su tutto S, si sceglie una distribuzione non uniforme in S che assegna una probabilità maggiore alla regione S_k. Il problema viene affrontato più ampiamente nel successivo paragrafo.

3.4.2 Adaptive Search

L'algoritmo Adaptive Search (AS) conserva il risultato di linearità del PAS generando i punti secondo una successione di distribuzioni di Boltzmann, come suggerisce Rubinstein in [61]. Una *distribuzione di Boltzmann* π_T ha densità di probabilità:

$$\pi_T(x) = \frac{e^{-\frac{f(x)}{T}}}{\int_S e^{-\frac{f(z)}{T}} dz} \tag{3.9}$$

dove T è un numero positivo e $x \in S$. Questa distribuzione di probabilità è molto interessante: per valori piccoli di T, π_T si concentra vicino al minimo globale. Vale infatti la seguente proposizione.

Teorema 3.4.1 (Romeijn e Smith, [60]). *Per ogni* $\varepsilon > 0$,

$$\lim_{T \downarrow 0} \pi_T(B_f(\varepsilon)) = 1, \tag{3.10}$$

dove $B_f(\varepsilon) = \{x \in S : |f(x) - f^*| < \varepsilon\}$.

Inoltre, per $T \to \infty$, π_∞ è equivalente alla distribuzione uniforme su S.
Lo schema dell'algoritmo Adaptive Search è il seguente:

Algoritmo 3.4.3 (Adaptive Search, AS).

siano $k \leftarrow 0$, $T_0 \leftarrow \infty$ e $f_0 \leftarrow \infty$;
loop
  si generi x secondo la distribuzione π_{T_k} in S;
  if $f(x) < f_k$
    $x_{k+1} \leftarrow x$; $f_{k+1} \leftarrow f(x_{k+1})$;
    $T_{k+1} \leftarrow \tau(f_{k+1})$;
    $k \leftarrow k + 1$;
  endif
forever
end

Secondo la convenzione, il parametro T prende il nome di *parametro di temperatura* o, più semplicemente, *temperatura*. Una scelta particolare dei parametri di temperatura $\{T_k\}_{k=0}^n$ è chiamata *cooling schedule*. In generale la cooling schedule è ottenuta mediante una funzione misurabile non crescente τ, definita nell'intervallo $[\min f(x), \max f(x)]$ e a valori reali positivi. La temperatura T_k è data da:

$$T_k = \tau(f(x_k)). \tag{3.11}$$

Il fatto che il campione sia prelevato dall'intero insieme S piuttosto che da sottoinsiemi annidati S_k, consente di ovviare alle difficoltà viste per il PAS. Il prezzo

da pagare consiste però nel fatto che la distribuzione che presiede all'estrazione del campione cambia durante l'esecuzione dell'algoritmo.

Nel caso generale, il numero di punti di prova campionati dall'algoritmo AS può essere influenzato da una scelta appropriata della cooling schedule, dove questa scelta dipenderà dalla forma della distribuzione π_T. In particolare, a ogni iterazione di AS, si vuole scegliere il parametro temperatura in modo tale che il numero di punti necessari per generare un punto in S sia piccolo.

La cooling schedule proposta da Romeijn e Smith in [59] per AS è la seguente: dato un punto y_k, migliore valore corrente della funzione, si sceglie la temperatura T_k in modo tale che la probabilità di generare un miglioramento usando come distribuzione π_{T_k} sia almeno $1-\alpha$, dove α è un numero positivo piccolo. In questo caso il numero atteso di prove ad ogni temperatura è al più $\frac{1}{1-\alpha}$, indipendentemente dalla dimensione n del problema. Di conseguenza, non solo il numero dei punti di un'iterazione di AS, ma anche il numero totale delle variabili aleatorie generate crescerà linearmente con la dimensione del problema.

Si noti che l'algoritmo Adaptive Search non è che una generalizzazione del PAS; infatti scegliendo $T_k = \infty$ a ogni iterazione, l'algoritmo Adaptive Search si riduce al PAS in cui ogni iterazione utilizza un metodo di accettazione-rifiuto; più precisamente quindi AS è una generalizzazione del PRS.
Inoltre è stato dimostrato (Romeijn e Smith, 1994, [59]) che il numero di iterazioni richieste AS è in genere inferiore a quello del PAS, con una limitazione superiore che dipende linearmente dalla dimensione del problema. Si veda a tal proposito il paragrafo 4.3.4.

Si può osservare che tutti i risultati forniti dai metodi Random Search non dipendono dalla scelta della classe delle distribuzioni di Boltzmann come distribuzioni di generazione. Infatti, questi risultati continuano a valere se le distribuzioni hanno una densità g_T avente la seguente forma più generale:

$$g_T(x) = h_T(f(x)), \tag{3.12}$$

dove la funzione h_T soddisfa l'ipotesi che $\frac{h_T(u)}{h_{T'}(u)}$ sia una funzione non decrescente in u per ogni $T \leq T'$ (Romeijn e Smith, 1992, [57]).

Visti gli ottimi risultati teorici ottenuti, si può concludere che se si trovasse il modo per generare punti direttamente dalla distribuzione di Boltzmann π_T, $\forall T > 0$, si sarebbe in grado di risolvere un'ampia classe di problemi. Per questa ragione, recentemente ci si è concentrati nel trovare algoritmi per generare in modo efficiente punti da distribuzioni arbitrarie usando un approccio legato alla teoria delle catene di Markov. Ciò significa che, invece di provare a generare punti da una distribuzione π in modo esatto, vengono costruite catene di Markov, in cui le transizioni sono generate "facilmente" e la cui distribuzione limite è equivalente alla distribuzione π. Esempi di metodi costruiti su questa idea sono l'algoritmo *Hit-and-Run* (cfr. [57], [7], [45]), l'algoritmo *Shake-and-Bake* (cfr. [10], [57], [58]) e gli algoritmi della classe *Random Walk* (cfr. [20], [19], [69]). Come si avrà modo

di osservare più avanti, anche alcune versioni dell'algoritmo Simulated Annealing seguono proprio questo approccio.

3.5 Simulated Annealing

Il Simulated Annealing ha origine dall'analogia tra il processo fisico di raffreddamento dei solidi, detto *annealing*, ed il problema di trovare soluzioni minime per i problemi di ottimizzazione nel discreto.

L'*annealing* denota un processo fisico in cui un solido immerso in un bagno caldo raggiunge uno stato di minima energia mediante un lento abbassamento della temperatura del sistema.

Il primo algoritmo di simulazione dell'annealing dei solidi è dovuto a Metropolis et al. (1953, [47]). Nel 1983, Kirkpatrick (cfr. [37]) utilizzò questo algoritmo per problemi di ottimizzazione combinatoria sostituendo l'energia con una funzione costo e gli stati del sistema fisico con le soluzioni di un problema di ottimizzazione. Per il suo successo in questo tipo di problemi (si vedano ad esempio i lavori di van Laarhoven e Aarts [39] e di Aarts e Korst [1]), lo studio del Simulated Annealing è stato esteso anche a problemi di ottimizzazione nel continuo. Per una trattazione generale nel continuo si veda [44].

A parte la motivazione puramente euristica sull'origine del metodo, la peculiarità del Simulated Annealing consiste nel fatto che l'algoritmo evita di stabilizzarsi in un minimo locale che non sia globale accettando, oltre che transizioni corrispondenti a un decremento del valore della funzione, anche transizioni che corrispondono a un incremento del valore della f; queste ultime avvengono però in modo limitato.

L'idea generale dell'algoritmo è quella di generare ad ogni prova un punto sulla base di una data distribuzione di probabilità D e la sua accettazione o rifiuto dipende da una funzione assegnata; tale funzione costituisce il cosiddetto *criterio di accettazione* e dipende da un parametro, chiamato *temperatura*, che controlla l'accettazione di quei punti a cui corrisponde un incremento del valore della funzione obiettivo. Al decrescere della temperatura decresce la probabilità di accettare tali transizioni.
Lo schema generale dell'algoritmo è il seguente:

Algoritmo 3.5.1 (Simulated Annealing, SA).

si determini il punto iniziale x_0 arbitrariamente in S;
$z_0 \leftarrow \{x_0\}$;
si determini $T_0 > 0$, valore iniziale della temperatura;
$k \leftarrow 0$;
loop
  si generi un punto y_{k+1} secondo una distribuzione $D(z_k; \cdot)$;
  si scelga p uniformemente in $[0, 1]$;
  if $p \leq A(x_k, y_{k+1}, T_k)$
    $x_{k+1} \leftarrow y_{k+1}$;

```
        z_{k+1} ← z_k ∪ {x_{k+1}};
      else
        x_{k+1} ← x_k;
        z_{k+1} ← z_k;
      end if
      T_{k+1} ← U(z_{k+1})
      k ← k + 1
    forever
end
```

$A(x_k, y_{k+1}, T_k)$ rappresenta il *criterio di accettazione* e generalmente viene scelto uguale al *criterio di Metropolis*:

$$A(x, y, T) = \min\left\{1, e^{-\frac{f(y)-f(x)}{T}}\right\}. \tag{3.13}$$

Tale funzione permette di accettare con una determinata probabilità punti, generati secondo la distribuzione $D(z_k; \cdot)$, a cui corrisponde un incremento del valore della funzione obiettivo. Si vede che tale probabilità tende a zero quando $T \to 0$.

Il vettore z_k costituisce in un certo senso la *memoria* dell'algoritmo, in quanto contiene tutta l'informazione prodotta fino alla k-esima iterazione.

Infine, $U(x_0, \ldots, x_{k+1})$ costituisce la cosiddetta *cooling schedule*, ossia una funzione a valori non negativi che permette di ridurre la temperatura.

L'algoritmo SA rappresenta un'applicazione approssimata dell'Adaptive Search descritto al paragrafo 3.4.2. In tale contesto si è avuto modo di osservare che AS è un algoritmo puramente concettuale, nel senso che risulta estremamente difficile generare i punti *direttamente* secondo la distribuzione di Boltzmann π_T. Poiché tale distribuzione gode della proprietà fondamentale data dal teorema 3.4.1, si può provare ad usare un approccio alternativo a quello *diretto*, nel modo seguente.
Si consideri una *passeggiata a caso* in S convergente alla distribuzione uniforme su S; sia $R(x; \cdot)$ la *probabilità di transizione* quando la catena di Markov si trova nello stato $x \in S$. "Filtriamo" ora la passeggiata a caso come segue: ad ogni iterazione, si generi un nuovo punto $y_{k+1} \in S$ a partire dal punto corrente x_k secondo la distribuzione $D(x_k; \cdot)$ e lo si accetti con probabilità $A(x_k, y_{k+1}, T_k)$; in questo caso la *probabilità di transizione* R è la probabilità di generare e accettare un nuovo punto a partire dal punto corrente (cfr. [12]). È stato dimostrato in [60] che la distribuzione della successione $\{X_k(T)\}_{k=0}^{\infty}$ generata in questo modo converge alla distribuzione di Boltzmann π_T, ossia che:

$$\lim_{k\to\infty} p(X_k(T) \in B_\varepsilon(f)) = \pi_T(B_\varepsilon(f)),\ \forall \varepsilon > 0. \tag{3.14}$$

Dall'equazione (3.10) del teorema 3.4.1, segue che:

$$\lim_{T\downarrow 0} \lim_{k\to\infty} p(X_k(T) \in B_\varepsilon(f)) = 1,\ \forall \varepsilon > 0. \tag{3.15}$$

Il Simulated Annealing si basa sull'equazione (3.15). La successione dei punti generati dall'algoritmo 3.5.1 costituisce quindi una realizzazione della catena di Markov $\{X_k(T)\}_{k=0}^{\infty}$, la cui convergenza all'ottimo globale è garantita quando la catena di Markov è omogenea (ossia la probabilità di transizione dallo stato x_k allo stato x_{k+1} non dipende dall'iterazione k) di lunghezza infinita per ogni valore della temperatura. Ovviamente, dal punto di vista pratico, l'implementazione di un simile algoritmo è impossibile; allora è indispensabile generare catene di Markov non omogenee, di lunghezza finita, per una successione, anch'essa finita, di valori decrescenti della temperatura T.

3.5.1 Convergenza

Recentemente sono stati pubblicati numerosi lavori sulla convergenza del Simulated Annealing. Lo scopo è quello di formulare opportune ipotesi in cui, se $\{x_k\}$ è la successione dei punti generati ed accettati dal SA, valga la relazione:

$$\lim_{k\to\infty} p[x_k \in B_\varepsilon(f)] = 1, \ \forall \varepsilon > 0, \tag{3.16}$$

oppure, se $\{y_k\}$ è la successione dei punti generati dal SA, valga la relazione:

$$\lim_{k\to\infty} p[y_k \in B_\varepsilon(f)] = 1, \ \forall \varepsilon > 0. \tag{3.17}$$

Riferiamoci inizialmente alla (3.16).
La condizione di base più spesso utilizzata in questo caso è la seguente:

$$\exists \varrho > 0 : \ \forall A \subseteq S, \ \forall z_k, \quad D(z_k, A) \geq \varrho \ mis(A); \tag{3.18}$$

cioè ad ogni iterazione del SA, ogni sottoinsieme dello spazio ammissibile ha una probabilità di essere raggiunto che risulta proporzionale alla sua misura di Lebesgue; in altri termini, il campionamento avviene su tutto S.

Il seguente teorema si basa sull'equazione (3.18) e garantisce la convergenza del SA indipendentemente dalla rapidità della cooling schedule (cfr. [8]).

Teorema 3.5.1 (Bélisle, 1992). *Sia $\{X_k(T)\}_{k=0}^{\infty}$ la successione generata dal Simulated Annealing. Si supponga che, per ogni punto iniziale x_0, si abbia:*

$$T_k \xrightarrow{p} 0, \quad \textit{per } k \to \infty.$$

Allora, la successione dei valori $\{f(X_k(T))\}_{k=0}^{\infty}$ converge in probabilità al minimo globale f^*; cioè, $\forall \varepsilon > 0$ e $\forall x_0$, si ha:

$$\lim_{k\to\infty} p[f(X_k(T)) - f^* < \varepsilon \ | X_0 = x_0] = 0.$$

Questo risultato sembra in contraddizione con un risultato fornito da Hajek (1988, [44]) per il SA applicato a problemi di ottimizzazione combinatoria. Hajek ha dimostrato che la successione $\{f(x_k)\}$ converge in probabilità a f^* se e solo se la

cooling schedule decresce in modo sufficientemente lento, ossia se la temperatura T_k viene scelta in modo tale che:

$$\sum_{k=1}^{\infty} e^{-\frac{d^*}{T_k}} = \infty,$$

con d^* massima distanza tra minimi locali che non siano globali.
L'apparente contraddizione tra i due risultati è dovuta al fatto che in ottimizzazione combinatoria il SA genera ogni nuovo punto in un intorno del punto corrente invece che su tutto l'insieme ammissibile, come nel caso continuo.

Un'immediata conseguenza del teorema 3.5.1 è data dal seguente corollario (cfr. [60]).

Corollario 3.5.1. *La successione* $f_k^* = \min_{0 \le i \le k} f(X_i)$ *generata dal Simulated Annealing converge al minimo globale* f^* *con probabilità 1, cioè:*

$$f_k^* \xrightarrow{q.c.} 0, \quad \textit{per } k \to \infty.$$

In [12], Dekkers e Aarts hanno formulato delle ipotesi che assicurano la validità della (3.15) inquadrando il SA nella teoria ergodica delle catene di Markov. Ad esempio, si richiede che $f : S \to \mathbb{R}$ sia uniformemente continua e che la distribuzione di probabilità, indicata con $g(x, y)$, soddisfi le seguenti:

- $mis(\Omega) > 0 \Rightarrow \int_{y \in \Omega} g(x, y)dy > 0, \quad \forall x \in S, \ \forall \Omega \subset S;$
- $g(x, y) = g(y, x), \quad \forall x, y \in S;$
- $g(x, y)$ non dipende dal parametro T.

In [18], Dorea ottiene gli stessi risultati di Dekkers per una distribuzione generica $g(x, y) = H_y$ indipendente da x.

In [42], Locatelli ha dimostrato la convergenza secondo la (3.16) di un algoritmo SA in cui i punti candidati vengono scelti in un intorno del punto corrente della catena di Markov, non su tutto lo spazio di ricerca. Come supporto della distribuzione $D(z_k; \cdot)$ si considera l'intersezione di S con la sfera $S(x_k, R)$, dove x_k è il punto corrente ed R è costante.

La cooling schedule viene scelta uguale a:

$$T_k = \begin{cases} f(x_k) - f_k^*, & \text{se } f(x_k) - f_k^* > \bar{\varepsilon}, \\ t_k, & \text{altrimenti}, \end{cases} \tag{3.19}$$

dove $\{t_k\}$ è una successione deterministica non crescente che converge a zero al tendere di k all'infinito ed $\bar{\varepsilon}$ è una costante tale che:

$$mis(S(x, R) \cap B_\varepsilon(f)) > 0, \ \forall x \in B_{2\bar{\varepsilon}}(f), \ \forall \varepsilon > 0.$$

L'idea è quella di accettare con una probabilità positiva punti a cui corrisponde un incremento del valore della funzione e far sì che tale probabilità decresca nel caso in cui il valore di f sia prossimo a f_k^*.
Sotto opportune ipotesi vale il seguente:

Teorema 3.5.2. *Sia N un intero sufficientemente grande e*

$$\Delta F = \max_{x \in S \setminus B_\varepsilon(f)} \max_{y \in S(x,R) \cap S} [f(y) - f(x)].$$

Se

$$T_k \geq (1+\mu) \left(\frac{N \Delta F}{\log k} \right), \ \mu > 0, \tag{3.20}$$

allora

$$\lim_{k \to \infty} p[x_k \in B_\varepsilon(f)] = 1, \ \forall \varepsilon > 0. \tag{3.21}$$

Osserviamo che sotto l'ipotesi (3.18) si può dimostrare la (3.16) ma è impossibile garantire la (3.17). Infatti, se $A \subseteq S \setminus B_\varepsilon(f)$ è tale che $mis(A) > 0$, la condizione (3.18) implica che:

$$\forall k, \quad p[y_k \in A] \geq \varrho \ mis(A) > 0.$$

In [43], vengono formulate delle ipotesi che garantiscono la (3.17) mediante un campionamento dei punti candidati y_k secondo una distribuzione avente supporto:

$$supp(D(z_k; \cdot)) = S(x_k, R_k) \cap S,$$

cioè il nuovo punto y_{k+1} appartiene ad un intorno del punto corrente x_k di ampiezza variabile. Si definisce:

$$R_k = \begin{cases} \delta > 0 & \text{se } f(x_k) - f_k^* > \bar{\varepsilon}, \\ r_k & \text{altrimenti,} \end{cases} \tag{3.22}$$

dove $\{r_k\}$ è una successione deterministica non crescente che converge a zero al tendere di k all'infinito. Come nel caso della T_k, l'idea di base è quella di avere un passo positivo che, vicino all'ottimo determinato fino alla k-esima iterazione, possa essere ridotto il più possibile, in modo da esplorare la regione in maniera più precisa.
Le ipotesi che garantiscono la convergenza per la successione $\{y_k\}$ richiedono ad esempio che:

1. $D(z_k; \cdot)$ coincida con la distribuzione uniforme su $S(x_k, R_k) \cap S$;
2. S sia compatto, convesso e a dimensione piena;
3. f sia continua e abbia un solo punto di minimo globale.

Si è osservato al paragrafo 3.4.2 che i risultati ottenuti per i metodi Random Search non dipendono strettamente dalla scelta della distribuzione di Boltzmann come distribuzione di generazione dei punti. Lo stesso discorso vale anche per i metodi della classe Simulated Annealing, per i quali i risultati dimostrati continuano a valere quando la densità di probabilità della distribuzione asintotica è data dalla (3.12).

3.5.2 Distribuzioni di probabilità e cooling schedule

Consideriamo ora il SA dal punto di vista pratico. L'efficienza del metodo dipenderà dalla scelta delle sue componenti fondamentali, ossia:

i) dalla temperatura iniziale T_0;

ii) dalla distribuzione di probabilità D;

iii) dalla cooling schedule U;

iv) dal criterio di stop.

Per quanto riguarda il valore iniziale della temperatura, un valore troppo alto di T_0 può aumentare notevolmente il tempo di calcolo mentre un valore troppo basso può non permettere l'accettazione delle transizioni corrispondenti a un incremento del valore della funzione obiettivo, con una maggiore probabilità di fallimento del metodo.

Come distribuzione di probabilità D viene solitamente scelta una distribuzione che consenta di generare facilmente punti in S, come ad esempio la *distribuzione uniforme* o alcune varianti che combinano il PRS con procedure di ricerca locale.

La scelta della cooling schedule U deve tener conto del fatto che una riduzione troppo rapida della temperatura può incrementare la probabilità di restare "intrappolati" in un minimo locale non globale; viceversa una sua riduzione troppo lenta può compromettere l'efficienza dell'algoritmo.

Infine, il criterio di stop deve consentire di effettuare un numero di prove sufficientemente elevato per ottenere una buona approssimazione della distribuzione di Boltzmann secondo l'equazione (3.15), in modo da garantire l'affidabilità del SA senza però perdere di vista la sua efficienza.

Recentemente, il SA è diventato oggetto di numerosi studi; si veda a tal proposito la bibliografia in [44]. Vediamo alcuni esempi di distribuzioni di probabilità e cooling schedule utilizzate in alcuni di questi lavori.

Distribuzioni di probabilità.

Sia x_k il punto corrente e y_{k+1} un nuovo punto candidato.
Un primo esempio è il seguente (Bohachevsky et al., 1986):

$$y_{k+1} = x_k + \Delta r \theta_k, \tag{3.23}$$

dove θ_k è un vettore aleatorio tale che $\|\theta_k\| = 1$ e Δr è un passo che dipende dalla funzione obiettivo e viene scelto in modo che il numero di transizioni "peggiori" accettate non superi il 60% di quelle totali.

Wang e Chen (1996) scelgono il passo Δr in modo *adattivo* in base alla distanza di x_k dal minimo globale noto o da una sua stima; la direzione θ_k non viene scelta uniformemente ma, se x_h è l'ultimo punto generato dal SA tale che $x_h \neq x_k,\ h < k$,

allora se $f(x_h) < f(x_k)$ viene favorita la direzione $x_h - x_k$, altrimenti si assegna probabilità maggiore alla direzione opposta.

Invece che considerare una ricerca di tipo *isotropico*, ossia in cui Δr è costante in ogni direzione, Vanderbilt e Louie (1984) sfruttano la struttura della funzione obiettivo attraverso le curve di livello, generando passi di lunghezza maggiore nelle direzioni in cui la f sembra variare più lentamente e passi molto piccoli quando la variazione è più rapida. Il punto candidato è scelto come segue:

$$y_{k+1} = x_k + Qu, \tag{3.24}$$

dove u è generato uniformemente in $[-\sqrt{3}, \sqrt{3}]^n$ e Q è la matrice ottenuta dalla fattorizzazione di Cholesky dell'inversa dell'hessiana di f, o meglio, di una sua approssimazione calcolata usando i punti campionati fino alla k-esima iterazione.

Corana et al. (1987) utilizzano una ricerca di tipo non isotropico modificando ad ogni iterazione una singola componente di x_k, secondo il seguente schema:

$$y_{k+1} = x_k + rv_{i+1}e_{i+1}, \tag{3.25}$$

dove r è scelto casualmente in $[-1, 1]$ e v_{i+1} è il passo di massima lunghezza consentito nella direzione canonica e_{i+1}. La scelta del passo dipende dal numero di punti accettati; all'aumentare di questi ultimi il passo viene incrementato, mentre viene ridotto nel caso opposto.

In Ingber (1989 e successivi) la scelta del punto candidato può essere effettuata:

a) in modo isotropico;

b) in modo non isotropico.

a) In questo primo caso (Ingber, 1989) si ha:

$$y_{k+1} = x_k + \Delta x_k, \tag{3.26}$$

dove Δx_k è scelto secondo la distribuzione normale:

$$g_k(\Delta x_k) = \frac{1}{(2\pi T_k)^{\frac{n}{2}}} e^{-\frac{\|\Delta x_k\|^2}{2T_k}} \tag{3.27}$$

o secondo la distribuzione di Cauchy:

$$g_k(\Delta x_k) = \frac{T_k}{(\|\Delta x_k\|^2 + T_k^2)^{\frac{n+1}{2}}}. \tag{3.28}$$

La convergenza degli algoritmi relativi è garantita quando T_k decresce più lentamente di $T_0/\log k$ se si usa la (3.27), o di T_0/k se invece si usa la (3.28). In quest'ultimo caso la temperatura decresce molto rapidamente e si parla di *fast annealing*.

b) Nel caso di ricerca non isotropica (Ingber, 1992), ogni componente Δx_k^i di Δx_k è scelta in $[-1, 1]$ in base a una diversa distribuzione avente densità:

$$g_k^i(\Delta x^i) = \frac{1}{2(|\Delta x^i| + T_k^i) \log\left(1 + \frac{1}{T_k^i}\right)}; \tag{3.29}$$

La successione di punti candidati è generata secondo lo schema seguente:

$$y_{k+1}^i = x_k^i + \Delta x_k^i (B_i - A_i), \tag{3.30}$$

dove $x_k^i \in [A_i, B_i]$.

Romeijn e Smith (1994) propongono di determinare il punto candidato usando il metodo *Hide-and-Seek*, ossia:

$$y_{k+1} = x_k + \lambda\theta_k, \tag{3.31}$$

dove θ_k è una direzione casuale tale che $\|\theta_k\| = 1$ e λ è un passo per cui $x_k + \lambda\theta_k \in S$. Questo modo di procedere può risultare inefficiente per effetto di un fenomeno detto *jamming*: infatti, quando il dominio di f è ad esempio un poliedro e x_k è molto vicino ad un angolo di S, il passo λ può risultare molto piccolo in gran parte delle direzioni θ_k; questo significa che y_{k+1} può essere scelto molto vicino a x_k, con un conseguente rallentamento dell'algoritmo. Per superare questo inconveniente, si introduce il concetto di *riflessione*, ossia:
si sceglie $X \supset S$ compatto; si genera un passo λ_k tale che $z_{k+1} = x_k + \lambda_k\theta_k \in X$; se $z_{k+1} \in S$ allora $y_{k+1} \leftarrow z_{k+1}$, altrimenti si determina $\tilde{z}_{k+1} = \partial S \cap [x_k, x_k + \lambda_k\theta_k]$, si sceglie una nuova direzione θ_k' e si pone $z_{k+1}' = \tilde{z}_{k+1} + \|z_{k+1} - \tilde{z}_{k+1}\|\theta_k'$; se $z_{k+1}' \in S$ allora $y_{k+1} \leftarrow z_{k+1}'$, altrimenti si ripete il ragionamento.

In Dekkers e Aarts (1991) e Ali e Storey (1996), il SA viene implementato come una catena di Markov di lunghezza L prefissata e distribuzione di generazione avente densità:

$$g(y; x) = \begin{cases} \frac{1}{mis(S)}, & \omega \leq \alpha, \\ \mathrm{LS}(S), & \omega > \alpha, \end{cases} \tag{3.32}$$

dove $mis(S)$ è la misura di Lebesgue di S, $\alpha \in (0, 1)$ prefissato, ω viene scelto a caso in $[0, 1]$ e LS è una procedura deterministica di ricerca locale, introdotta per "accelerare" l'individuazione dell'ottimo globale.

Ali et al. (2002) usano lo stesso procedimento di Dekkers e Aarts campionando però un insieme finito, detto *popolazione*, iniziale di punti; ad ogni iterazione i punti in cui la funzione obiettivo è più alta vengono sostituiti con nuovi punti ottenuti tramite ricerche locali, fino a quando l'insieme dei punti si concentra in un intorno di un punto di minimo globale di f.

Similmente all'approccio di Dekkers e Aarts, Desai e Patil (1996) determinano i nuovi punti candidati applicando una ricerca locale ad una perturbazione di un minimo locale x_k; in questo modo la successione prodotta dall'algoritmo diventa una successione di minimi locali.

In [46], Lucidi e Piccioni scelgono il punto candidato uniformemente in S e utilizzano il criterio di Metropolis non per decidere se accettare o meno y_{k+1}, ma per decidere se applicare o meno una ricerca locale. Nella funzione (3.13), $f(x_k)$ viene sostituito con l'ottimo corrente f_k^*, in modo che si applichi sicuramente la ricerca locale quando $f(y_{k+1}) < f_k^*$ (in tal caso si parla di *tunneling*), ma non se ne escluda l'applicazione nel caso contrario.

Cooling schedule

Bohachevsky et al. (1986) mirano ad accettare con bassa probabilità le transizioni verso valori di f più alti del valore corrente quando x_k è vicino ad una stima dell'ottimo globale; la cooling schedule scelta con questo scopo è la seguente:

$$U(z_k) = \beta[f(x_k) - \hat{f}^*]^{g_1}, \tag{3.33}$$

dove β e g_1 sono costanti positive e $\hat{f}^*$ è una stima di f^*.

In Vanderbilt e Louie (1984), la temperatura viene ridotta ogni M iterazioni di un fattore $\chi_T \in (0,1)$:

$$U(z_k) = \chi_T^{\left\lfloor \frac{k}{M} \right\rfloor} T_0. \tag{3.34}$$

La scelta di T_0 avviene sulla base della varianza di un campione casuale su f, in modo che la cooling schedule tenga conto della "scala" della funzione.

In caso di ricerca non isotropica, Ingber (1992 e successivi) opera una riduzione della temperatura con velocità esponenziale in ogni direzione:

$$T_k^i = T_0^i e^{-c_i k^{\frac{1}{n}}}, \tag{3.35}$$

dove T_0^i e c_i sono costanti opportune.
In (Ingber, 1989) si considera anche la possibilità di aumentare periodicamente la temperatura in base all'andamento della f in ogni direzione, cioè definiti:

$$s_i = \frac{\partial f(x_k^*)}{\partial x_i}, \quad \text{e} \quad s_{\max} = \max_{1 \le i \le n} s_i,$$

con x_k^* ottimo corrente di f, il nuovo valore di T_k^i è dato dalla seguente:

$$\overline{T}_k^i = T_k^i \left(\frac{s_{\max}}{s_i} \right). \tag{3.36}$$

Questo modo di procedere, detto *re-annealing*, consente di far "ripartire" l'algoritmo quando la funzione varia molto lentamente in una direzione particolare, pur non essendo ancora soddisfatto un opportuno criterio di stop.
Secondo Ingber (1993), un'altra possibile strategia per superare l'inconveniente di una convergenza molto lenta dell'algoritmo, è quella di usare una tecnica di tipo "Multistart", effettuando una molteplicità di prove con punti iniziali differenti e

introducendo il concetto di *quenching* (riduzione molto rapida della temperatura) secondo lo schema:

$$T_k^i = T_0^i e^{-c_i k^{\frac{Q_i}{n}}},\ Q_i > 1. \tag{3.37}$$

Nonostante dal punto di vista empirico l'algoritmo fornisca risultati spesso molto migliori rispetto al SA classico (cfr. [35] e [36]), si ha lo svantaggio di perdere la garanzia della convergenza al minimo globale.

Criteri di stop

Alcune delle condizioni più comuni per l'arresto dell'algoritmo sono le seguenti:

1. viene superato un numero prefissato di iterazioni in cui non si accettano nuovi punti;
2. il numero di accettazioni in rapporto ai punti generati è inferiore a un valore prestabilito;
3. si ha una lenta variazione dei valori della funzione lungo una direzione prefissata o della media di tali valori per un numero fissato di iterazioni;
4. si raggiunge una stima del minimo globale entro un'accuratezza prefissata.

In ogni caso, l'idea generale è quella di interrompere l'esecuzione dell'algoritmo quando non si hanno evidenti miglioramenti dell'ottimo corrente dopo un numero sufficientemente grande di iterazioni. Ovviamente la scelta dei criteri di stop è strettamente legata alla distribuzione di generazione dei punti e alla cooling schedule.

3.5.3 Equazione di Langevin

Una classe di metodi molto simili al Simulated Annealing è quella degli algoritmi basati sull'*equazione di Langevin* in $\mathbb{R}^n$ (cfr. [62]):

$$d(x(t)) = -\nabla f(x(t))dt + \sqrt{2T(t)}d\omega(t), \tag{3.38}$$

dove ∇f è il gradiente di f, $T(t)$ la temperatura all'istante $t \in [0, \infty)$ e $\omega(t)$ è il moto browniano standard in $\mathbb{R}^n$.
L'equazione (3.38) è la generalizzazione della legge del moto browniano di una particella in moto in un fluido viscoso; al di là del suo significato fisico, questa equazione rappresenta, nel contesto degli algoritmi stocastici, la legge del moto di un punto in $\mathbb{R}^n$ soggetto a due componenti: una lungo la direzione di discesa corrispondente all'antigradiente $-\nabla f(x)$ e l'altra lungo una direzione casuale dipendente dal parametro temperatura $T(t)$.

L'algoritmo che deriva dall'equazione (3.38) consiste nel simulare un "processo di decisione di Markov" (cfr. [11], pagg. 273-314) in cui, supponendo che $T(t) \to 0$ quando $t \to \infty$,

$$x_{k+1} = x_k - a_k \nabla f(x_k) + bW_k,\ k = 0, 1, \ldots$$

dove $\{W_k\}$ è una successione di variabili aleatorie indipendenti e identicamente distribuite, con distribuzione normale standard, $a_k = \Delta t_k$ e $b_k = \sqrt{2a_k T(t_k)}$ (cfr. [62]).

L'algoritmo di Langevin, come il SA, è un metodo stocastico di discesa che prevede l'accettazione di valori "peggiori" rispetto all'ottimo corrente, in modo da poter uscire dalla regione di attrazione dei minimi locali non globali. La stretta correlazione col SA è dovuta al fatto che il processo di Markov è costruito in modo tale che la distribuzione asintotica sia ancora la distribuzione di Boltzmann, $\forall T > 0$. Infatti, se $\sqrt{2T(t)} = \text{cost} = \varepsilon$, allora la densità di probabilità del processo $x^{\varepsilon}(t)$ soluzione di (3.38), per $t \to \infty$ e a meno di una costante, tende alla distribuzione di Boltzmann di temperatura $\varepsilon^2/2$:

$$e^{\frac{2f(x_0)}{\varepsilon^2}}. \tag{3.39}$$

Nonostante molti algoritmi di ottimizzazione siano basati su questo approccio (si veda la bibliografia in [62]), pochi di questi sono implementabili. Un esempio è l'algoritmo proposto da Aluffi-Pentini et al. (cfr. [3]), in cui si sfrutta l'idea di integrare numericamente l'equazione (3.38) per seguire le traiettorie dell'equazione differenziale stocastica; usando il metodo di discretizzazione di Eulero-Cauchy, la soluzione viene approssimata utilizzando la seguente equazione alle differenze finite:

$$\begin{cases} x(t_{k+1}) &= x(t_k) - \Delta t_k \nabla f(x(t_k)) + \sqrt{2T(t_k)}(\omega_{k+1} - \omega_k), \\ x(t_0) &= x_0. \end{cases} \tag{3.40}$$

Dal punto di vista pratico, l'algoritmo fornisce risultati complessivamente accettabili, anche se il numero di valutazioni di funzione è piuttosto elevato.

3.6 Metodi Random Directions

Una strategia per individuare il minimo globale di una funzione consiste nel generare una successione di punti corrispondenti a valori decrescenti della f e convergente ad un punto di minimo globale. Una simile tecnica prende il nome di *Global Decrease*. Se poi, in alternativa al campionamento casuale dei punti in S si utilizzano *direzioni casuali*, i metodi in questione prendono il nome di metodi *Random Directions*.

Sulla base dello schema studiato da Solis e Wets in [65], alla $k+1$-esima iterazione viene scelta una funzione di distribuzione F, in genere funzione del punto corrente x_k, secondo la quale viene generato un vettore di prova ξ non necessariamente appartenente all'insieme ammissibile. Il nuovo punto si ottiene mediante l'applicazione di un'opportuna trasformazione:

$$D : S \times \mathbb{R}^n \to S.$$

La particolare scelta della distribuzione F e dell'applicazione D consente di ottenere un'ampia varietà di metodi; ad esempio, se F è la distribuzione uniforme su S e $D(x,\xi) = \arg\min\{f(x), f(\xi)\}$, il metodo si riduce al PRS mentre, se $D(x,\xi) = \mathcal{L}(\xi)$, dove $\mathcal{L}$ rappresenta una procedura di ricerca locale, si riottiene il Multistart.

Algoritmo 3.6.1 (Random Directions).

sia $k \leftarrow 0$ e $D : S \times \mathbb{R}^n \to S$;
si scelga $x_0 \in S$;
sia $f^* \leftarrow f(x_0)$;
loop
  sia F una distribuzione di probabilità e si generi un vettore aleatorio ξ secondo la distribuzione F;
  $x_{k+1} \leftarrow D(x_k, \xi)$;
  $f^* \leftarrow \min(f^*, f(x_{k+1}))$;
  $k \leftarrow k + 1$;
forever
end

In alcune varianti dell'algoritmo di base, la F è scelta come la distribuzione uniforme su una sfera di raggio fissato centrata in x_k; inoltre, poiché x_k e ξ individuano una retta aleatoria, D può essere scelta come una sorta di ricerca lineare lungo la direzione $\xi - x_k$; in particolare, nella versione proposta da Gaviano in [22], D è data da:

$$D(x_k, \xi) = x_k + \alpha^*(\xi - x)$$

$$\alpha^* = \arg\min_{\alpha} f(x_k + \alpha(\xi - x_k)),$$

dove $\alpha \in [0, 1]$. L'intervallo di variabilità di α può essere esteso ad uno tra i seguenti intervalli:

$$[0, 2], \quad [0, +\infty),$$

oppure limitato all'insieme discreto $\{0, 1\}$ (cfr. [62]).

I metodi Random Directions come quello appena descritto, combinano quindi una strategia probabilistica, la scelta caso della direzione, con una deterministica, nel caso specifico la minimizzazione lungo una direzione.

Per quanto riguarda la convergenza, si dimostra il seguente risultato (cfr. [22], [65]).

Teorema 3.6.1. *Sia $\{x_k\}$ la successione dei punti generati dall'algoritmo 3.6.1 e sia $x^* \in B_\varepsilon(f)$. Si supponga che valgano le seguenti ipotesi:*

i) D sia tale che $f(D(x,\xi)) \leq f(x)$ (cioè non vengono accettati passi che portano ad un peggioramento dell'ottimo corrente);

ii) per ogni insieme di Borel B avente misura di Lebesgue positiva e denotata con p_k la probabilità associata a B alla k-esima iterazione, si abbia:

$$\prod_{k=1}^{\infty}(1 - p_k(B)) = 0$$

(cioè si evita la possibilità di trascurare insiemi di Borel di misura positiva al crescere del numero di iterazioni);

allora:

$$\lim_{k\to\infty} p(x_k \in B_\varepsilon(f)) = 1, \quad \forall \varepsilon > 0.$$

Infatti:

$$\lim_{k\to\infty} p(x_k \in B_\varepsilon(f)) = \lim_{k\to\infty} (1 - p(x_k \notin B_\varepsilon(f))) = 1 - \lim_{k\to\infty} \prod_{i=1}^{k}(1 - p_i(B_\varepsilon(f))),$$

che tende a 1 per l'ipotesi ii).

I metodi Random Directions risultano generalmente abbastanza semplici da implementare ma, non sfruttando procedure di ricerca locale, sembrano adatti solo per classi particolari di problemi, aventi ad esempio pochi minimi locali e per i quali non si dispone di informazioni sulle derivate della funzione obiettivo. Inoltre è molto difficile individuare opportuni criteri di stop.

Osservazione 3.6.1. Un esempio di algoritmi Random Directions è la classe dei metodi stocastici basati sull'equazione di Langevin (3.38). Come si è osservato al paragrafo 3.5.3, questi metodi sfruttano la traiettoria $x(t)$, soluzione dell'equazione (3.38), che sotto opportune ipotesi converge al minimo globale in senso probabilistico (cfr. [56]*).*

3.7 Improvement of Local Minima

L'idea di introdurre procedure di ricerca locale in tecniche Global Decrease, conduce a una classe di metodi che generano una successione decrescente di minimi locali; idealmente, l'ultimo minimo locale ottenuto dovrebbe essere quello globale. Questo approccio viene detto *Improvement of Local Minima* (ILM).

Esistono esempi di metodi deterministici appartenenti a questa classe che usano perturbazioni della funzione obiettivo, come il metodo di *tunneling* (Levi e Montalvo [41], Levi e Gomez [40], Barhen et al. [5]) e i metodi *filled function* (Ge, [29]). Alcune versioni del tunneling inglobano anche elementi stocastici, come il *random tunneling* di Lucidi citato al paragrafo 3.5.2 (cfr. [46]) o l'*SPT* di Oblow (cfr. [50]). Alla classe ILM appartengono anche i metodi Random Directions che utilizzano ricerche locali, di cui ora analizziamo alcuni esempi.

In [2], Abaffy et al. studiano due algoritmi del tipo ILM per i quali vengono stabiliti dei criteri di stop di tipo bayesiano.
Gli algoritmi proposti sono casi particolari del seguente schema generale:

Algoritmo 3.7.1 (Improvement of Local Minima, ILM).

siano $G_i : S \to S$ e $F_i : S \to S$ particolari procedure da specificare per ogni algoritmo;
si scelga $x_0 \in S$, arbitraria approssimazione di $x^* \in S^*$;
$i \leftarrow 0$;
loop
si calcoli $y \in S$ secondo la procedura G_i :

$$y \leftarrow G_i(x_i);$$

si calcoli $x_{i+1} \in S$ secondo la procedura F_i :

$$x_{i+1} \leftarrow F_i(y);$$

$i \leftarrow i + 1$;
forever
end

Si indichi ora con Γ una procedura deterministica per l'individuazione del minimo globale in una direzione, ossia un algoritmo che, dato un punto x, una direzione h e la funzione $f(x + \alpha h)$, determini un $\bar{\alpha} \in \mathbb{R}^+$ tale che:

$$f(x + \bar{\alpha}h) \leq f(x + \alpha h), \quad \forall \alpha \geq 0.$$

Sia inoltre Φ una procedura di ricerca locale deterministica che, dato un punto x, individua un minimo locale x' tale che $f(x') \leq f(x)$.
Gli algoritmi studiati sono i seguenti:

Algoritmo 3.7.2 (Improvement of Local Minima, Caso 1).

sia $\Gamma : S \to S$ una procedura di minimizzazione globale unidimensionale;
sia $\Phi : S \to S$ una procedura di minimizzazione locale;
sia $\bar{x} \in S$ fissato;
si scelga $x_0 \in S$, arbitraria approssimazione di $x^* \in S^*$;
$i \leftarrow 0$;
loop
repeat
si scelga $h \in \mathbb{R}^n$ uniformemente su una sfera B di centro $\bar{x}$ e raggio 1;
tramite Γ si calcoli $\bar{\alpha}$ tale che:

$$f(\bar{x} + \bar{\alpha}h) = \text{min globale di } f(\bar{x} + \alpha h), \ \alpha \geq 0;$$

$y \leftarrow \bar{x} + \bar{\alpha}h$;
until $f(y) < f(x_i)$
$x_{i+1} \leftarrow \Phi(y)$;
$i \leftarrow i + 1$;
forever
end

Algoritmo 3.7.3 (Improvement of Local Minima, Caso 2).

sia $\Phi : S \to S$ una procedura di minimizzazione locale;
sia $\bar{x} \in S$ fissato;
si scelga $x_0 \in S$, arbitraria approssimazione di $x^* \in S^*$;
$i \leftarrow 0$;
loop
  repeat
    si scelga $h \in \mathbb{R}^n$ uniformemente su una sfera B di centro $\bar{x}$ e raggio 1;
    si scelga un passo $\alpha \geq 0$ tale che $\bar{x} + \alpha h \in S$;
    $y \leftarrow \bar{x} + \alpha h$;
  until $f(y) < f(x_i)$
  $x_{i+1} \leftarrow \Phi(y)$;
  $i \leftarrow i + 1$;
forever
end

L'applicazione degli algoritmi 3.7.2 e 3.7.3 per un numero di iterazioni $m \to \infty$ genera una *traiettoria*, ossia una successione decrescente di minimi locali di cui l'ultimo individuato è quello globale.

Si dimostra il seguente teorema di convergenza.

Teorema 3.7.1. *Se f è una funzione continua, gli algoritmi 3.7.2 e 3.7.3 convergono in probabilità al minimo globale, ossia la successione $\{x_k\}$ dei punti generati soddisfa la seguente:*

$$\lim_{k \to \infty} p(x_k \in B_\varepsilon(f)) = 1.$$

La dimostrazione segue immediatamente dal fatto che se f è continua, la misura di $B_\varepsilon(f)$ è positiva $\forall \varepsilon > 0$.

I metodi della classe ILM sono considerati molto rapidi nella risoluzione di un problema di ottimizzazione globale ma svantaggiosi per la mancanza di criteri di stop adeguati.

Questo problema viene affrontato in [2] tramite un'analisi statistica fondata sul teorema di Bayes: il procedimento adottato consente di scrivere la distribuzione "a posteriori" delle probabilità di saltar fuori dalle regioni di attrazione dei minimi locali, avendo individuato un minimo inferiore a quello corrente; questa distribuzione permette di formulare alcuni criteri di stop per gli algoritmi 3.7.2 e 3.7.3.

Prima di enunciare i criteri proposti, è necessario introdurre alcune notazioni. Supponendo di conoscere il numero k dei minimi locali di f, o almeno una sua stima, si indica con θ_j la probabilità di uscire dalla regione di attrazione del j-esimo minimo locale. Se l'algoritmo 3.7.2 o 3.7.3 individua τ differenti minimi locali in m prove, allora si indica con n_τ il numero di iterazioni prima di spostarsi dall'ultimo minimo trovato e con E_j il valor medio a posteriori di $\theta_j, j = 1, \ldots, \tau$. Inoltre, posto $\alpha = n_\tau/m$ e $P = (1 - E_\tau)^{n_\tau}$, $\alpha/(\alpha + (1 - \alpha)P)$ rappresenta la probabilità

che l'ultimo minimo individuato sia globale nell'ipotesi che in n_τ iterazioni non siano stati individuati minimi inferiori.
Con $\hat{\theta}_j$ si indica il valore atteso di $\bar{\theta}_j$, probabilità di lasciare un minimo locale nell'ipotesi che siano stati trovati j minimi, mentre $\mathbf{t}$ è una variabile aleatoria che rappresenta il numero di minimi locali individuabili in m ricerche lineari; il valor medio "a priori" di $\mathbf{t}$ è:

$$E_m(\mathbf{t}) = \sum_{t=0}^{\min(m,k)} t \sum_{n_0+\ldots+n_t=m} (1-\hat{\theta}_0)^{n_0-1}\hat{\theta}_0 \cdots (1-\hat{\theta}_{t-1})^{n_{t-1}-1}\hat{\theta}_{t-1}(1-\hat{\theta}_t)^{n_t},$$

mentre il valor medio "a posteriori" di $\mathbf{t}$ è:

$$S_m = \sum_{t=0}^{\tau-1} t \sum_{n_0+\ldots+n_t=m} (1-\hat{\theta}_0)^{n_0-1}\hat{\theta}_0 \cdots (1-\hat{\theta}_{t-1})^{n_{t-1}-1}\hat{\theta}_{t-1}(1-\hat{\theta}_t)^{n_t}.$$

Se l'ultimo minimo trovato è quello globale, si dimostra che:

$$\lim_{m\to\infty} S_m = 0.$$

Secondo la teoria elaborata da Abaffy et al., l'esecuzione dell'algoritmo termina se è soddisfatto uno dei seguenti criteri:

criterio I:
$$E_\tau < \varepsilon, \quad \text{con} \quad \varepsilon > 0 \text{ fissato;} \tag{3.41}$$

criterio II:
$$1-(1-E_\tau)^{n_\tau} < \varepsilon, \quad \text{con} \quad \varepsilon \in (0,1) \text{ fissato;} \tag{3.42}$$

criterio III:
$$\left(1-\frac{\alpha}{\alpha+(1-\alpha)P}\right) < \varepsilon, \quad \text{con} \quad \varepsilon > 0 \text{ fissato;} \tag{3.43}$$

criterio IV:
$$S_m < \varepsilon, \quad \text{con} \quad \varepsilon > 0 \text{ fissato;} \tag{3.44}$$

criterio V:
$$(E_m(\mathbf{t}) - \tau) < \varepsilon, \quad \text{con} \quad \varepsilon > 0 \text{ fissato;} \tag{3.45}$$

In altri termini, l'algoritmo si arresta se è sufficientemente piccola la probabilità di individuare un minimo inferiore al τ-esimo, oppure se tende a 1 la probabilità che non esista un minimo inferiore al τ-esimo non avendo ottenuto miglioramenti con le ultime n_τ iterazioni.

L'effettiva applicabilità dei criteri (3.41)-(3.45) è stata successivamente verificata in [25].

Capitolo 4

Complessità computazionale

4.1 Introduzione

Come si è visto nei capitoli precedenti, lo scopo dell'ottimizzazione globale è quello di individuare almeno un punto di ottimo della funzione obiettivo, ovviamente quando questo esiste. Tipicamente questo scopo viene raggiunto mediante lo studio di metodi basati essenzialmente sulle proprietà della funzione da minimizzare e dell'insieme ammissibile.

Generalmente, i metodi di risoluzione adottati devono poter essere applicati anche quando i dati del problema variano. Non è però pensabile la costruzione di un metodo che valga per qualsiasi problema di ottimizzazione, in quanto un simile metodo non potrebbe sfruttare le caratteristiche particolari di f o di S, risultando quindi inefficiente per il problema specifico. È però possibile identificare classi di problemi molto simili tra loro, in maniera tale da progettare metodi in grado di risolvere tutti i problemi della classe nel modo più efficiente possibile.

La risoluzione del problema avviene di norma mediante l'utilizzo di un calcolatore; questo significa che i metodi sviluppati devono essere di tipo algoritmico, ossia devono operare entro un insieme ben definito di regole. Ogni algoritmo, quando eseguito, richiede un costo in termini di risorse di tempo e di spazio che prende il nome di *complessità computazionale dell'algoritmo.* Conoscere il *costo* degli algoritmi è molto interessante perché, ad esempio, consente di poter confrontare le loro prestazioni.
Inoltre in generale la teoria della complessità computazionale si occupa non solo di valutare il costo di un algoritmo, ma anche di capire se tale costo sia dovuto solo alla struttura dell'algoritmo utilizzato oppure anche alle proprietà intrinseche

alla classe di problemi per cui l'algoritmo è stato ideato; in questo caso si parla di *complessità computazionale del problema.* Anche questo aspetto è estremamente utile in quanto consente, almeno per alcune classi di problemi, di conoscere a priori informazioni sulla complessità di un possibile algoritmo, informazioni che possono essere utilizzate per progettare algoritmi efficienti. In questo capitolo ci riferiremo in particolare alla complessità computazionale di un algoritmo.

Osserviamo che gli algoritmi proposti per la risoluzione del problema 1.1.1 non sono affatto semplici. Un algoritmo è spesso combinazione di diverse procedure che vengono richiamate ad iterazioni non prevedibili a priori. Inoltre un algoritmo può dipendere da parametri che devono essere precisati in modo tale da rendere l'algoritmo efficiente.
Queste poche osservazioni danno un'idea di quanto complicato possa essere lo studio della complessità di un algoritmo.

4.2 Principali definizioni

Per poter valutare la complessità computazionale di un algoritmo è necessario preliminarmente definire i concetti di *problema*, di *algoritmo* e di *risorsa* (cfr. [9]).

Un *problema computazionale* è un quesito di carattere generale che, solitamente, dipende da parametri e variabili di valori non specificati. Una descrizione non ambigua del problema richiede la conoscenza di tutti i parametri; l'assegnazione di particolari valori ai parametri dà luogo a quella che viene chiamata *istanza* del problema.

Definizione 4.2.1. *Un problema computazionale è l'insieme di tutte le istanze di un problema.*

Un *algoritmo*, definito come una successione finita e non ambigua di passi computazionali, è *corretto* se risolve il problema, cioè se genera una soluzione per ogni istanza del problema. La valutazione della sua complessità dipende essenzialmente dalle risorse utilizzate, dal criterio utilizzato per la misurazione del *costo* associato ad ogni risorsa e dal *modello di calcolo* adottato.

Le *risorse* coinvolte nel calcolo del costo di un algoritmo sono quelle di *spazio* (massima quantità di memoria usata per immagazzinare i risultati parziali del procedimento necessari per le operazioni successive) e di *tempo* (durata complessiva del calcolo). Il *costo* in termini di tempo e/o di spazio viene valutato rispetto al comportamento asintotico della *dimensione* dell'istanza, dove per *dimensione* si intende il numero di simboli utilizzati per la codifica dell'istanza stessa. In genere, ad una crescita della dimensione corrisponde infatti una crescita del costo dell'algoritmo. Nel seguito, per *costo* di un algoritmo intenderemo il numero totale delle operazioni richieste dall'algoritmo per generare una soluzione del problema; questo corrisponde a scegliere come risorsa di riferimento il tempo di calcolo piuttosto che la quantità di memoria, vista la maggiore importanza della prima nella pratica.

Un criterio di classificazione dei problemi computazionali si basa sulla definizione di algoritmo efficiente.

Definizione 4.2.2. *Un algoritmo è detto* efficiente *quando il suo costo computazionale ha una crescita polinomiale nella dimensione dell'input.*

Dalla precedente definizione discende la seguente:

Definizione 4.2.3. *Un problema computazionale è detto:*

- intrattabile*, se può essere dimostrato che non esistono algoritmi di risoluzione di tipo polinomiale;*
- presumibilmente intrattabile *se da un lato non si conoscono algoritmi polinomiali che lo risolvono, dall'altro non ne è stata dimostrata la non esistenza;*
- trattabile*, se esiste un algoritmo polinomiale per la sua risoluzione.*

Un algoritmo, seppur efficiente secondo la definizione 4.2.2, potrebbe non essere applicabile nella pratica se il grado del polinomio è molto alto, anche se, in generale, questa circostanza non si verifica.

4.2.1 Modelli di calcolo

L'applicazione puramente teorica di un algoritmo numerico fornisce un risultato non affetto da errore. Al contrario, quando l'algoritmo viene eseguito su un calcolatore, il risultato dipende, oltre che da fattori non prevedibili che danno luogo ai cosiddetti *errori accidentali*, anche dal particolare sistema *floating point* utilizzato dal calcolatore per la rappresentazione numerica. Ciò significa che uno stesso algoritmo eseguito su calcolatori differenti può fornire risultati diversi. Perché abbia senso confrontare gli algoritmi risulta quindi indispensabile l'introduzione di *modelli di calcolo*, che consentano di dare una definizione formale di algoritmo valida nel contesto generale. I modelli di calcolo usati più frequentemente sono:

1. la Macchina di Turing,
2. il modello algebrico (o reale),
3. il modello "black-box".

Vediamoli più in dettaglio.

1. La Macchina di Turing (MdT) costituisce un modello di calcolo astratto utilizzato per individuare i passi di una computazione. Nonostante la sua struttura molto elementare, la MdT è in grado di eseguire, almeno dal punto di vista teorico, ogni calcolo eseguibile su macchine reali senza perdita di efficienza. Ne consegue che ha senso prendere in considerazione solo algoritmi che possono essere eseguiti su una MdT.

La MdT può essere rappresentata come un insieme di k nastri su cui agiscono altrettanti dispositivi mobili di lettura e/o scrittura detti *testine*. Ogni nastro, finito a destra ed infinito a sinistra, è suddiviso in celle ciascuna delle quali può contenere un solo simbolo dell'alfabeto per volta. Solitamente i nastri sono in numero $k \geq 3$, dove il primo nastro è utilizzato per scrivere l'input, l'ultimo per leggere l'output e i rimanenti costituiscono nastri di lavoro.

Per come è definita, la MdT è in grado di operare solo su numeri interi o razionali; la complessità di un algoritmo viene valutata sulla base della dimensione dell'input.

Nonostante la stretta correlazione col calcolatore su cui gli algoritmi vengono implementati, il modello di Turing presenta alcuni importanti inconvenienti:

- la complessità di un algoritmo non dipende con continuità dai dati del problema: è possibile dimostrare che anche se i dati di input di due problemi sono molto vicini tra loro, lo spazio necessario alla loro memorizzazione può risultare molto diverso;
- solitamente gli algoritmi presi in considerazione sono piuttosto complessi; questo si ripercuote inevitabilmente sulla valutazione della loro complessità computazionale quando si deve tener conto degli effetti degli arrotondamenti.

2. Il modello algebrico (o a numeri reali) consente di operare in aritmetica esatta, essendo possibile la codifica dei numeri reali. Questo significa che in questo modello valgono tutti i concetti introdotti per gli spazi metrici, come quelli di convergenza, continuità, punto limite e velocità di convergenza, e tutte le proprietà ad essi correlate.

3. Come il modello algebrico, anche il modello "black-box" permette di operare su numeri reali. Inoltre, non essendo generalmente nota alcuna informazione a priori sul *costo* della funzione o della sua derivata in un punto particolare, quando questa è richiesta, tale modello consente di assegnare i valori della f e delle sue derivate mediante delle funzioni definite dall'utente. Tale modello può essere paragonato ad una MdT se si suppone l'esistenza di un ulteriore nastro da cui l'informazione aggiuntiva possa essere letta.

4.2.2 Classi di complessità

La teoria della complessità computazionale si basa essenzialmente sul concetto di *algoritmo polinomiale* e di *problema decisionale*, nonché sul concetto di *riducibilità* di un problema decisionale L ad un problema L^* e su quello di *algoritmo non deterministico*. Mediante queste nozioni sarà possibile definire le fondamentali classi di complessità per i problemi decisionali, ossia le classi *P*, *NP*, *NP-complete* e *NP-hard*. Di seguito vengono definiti in modo conciso i concetti fondamentali, prediligendo l'aspetto intuitivo piuttosto che quello assolutamente rigoroso.

Definizione 4.2.4 (Algoritmo polinomiale). *Sia $t(i)$ il tempo di esecuzione di un algoritmo relativamente all'input i e sia $\hat{t}(n) = \max_{i:|i|\leq n} t(i)$ il tempo relativo al caso peggiore, essendo $|i|$ la dimensione di i. Un algoritmo per il quale esiste un polinomio p tale che $\hat{t}(n) \in O(p)$, è detto* di costo in tempo polinomiale *o, più semplicemente,* polinomiale. *In caso contrario l'algoritmo è detto* di costo in tempo esponenziale.

Ricordiamo che in generale $O(h)$ è l'insieme delle funzioni che asintoticamente crescono meno velocemente di h, quindi dire che $f \in O(h)$ significa stabilire una limitazione superiore alla crescita di f. Nel caso della definizione 4.2.4, un algoritmo è polinomiale se il tempo di esecuzione risulta limitato superiormente da un polinomio in n.
Un algoritmo ha invece *costo in tempo esponenziale* se esiste una costante C tale che $\hat{t}(n) \in O(2^{n^C})$.
Gli algoritmi polinomiali si possono classificare ulteriormente in:

- costanti: se esiste una costante C tale che $\hat{t}(n) \leq C,\ \forall n$;
- logaritmici: se $\hat{t}(n) \in O(\log n)$;
- polilogaritmici: se esiste una costante C tale che $\hat{t}(n) \in O(\log^C n)$;
- lineari: se $\hat{t}(n) \in O(n)$;
- quadratici: se $\hat{t}(n) \in O(n^2)$.

Poiché ogni algoritmo appartenente a una delle categorie dell'elenco appartiene anche a quelle successive, si definisce *complessità* di un algoritmo la più piccola classe di complessità a cui l'algoritmo appartiene rispetto, ad esempio, alla risorsa tempo.

Lo studio della complessità riguarda i cosiddetti *problemi decisionali.*

Definizione 4.2.5 (Problema decisionale). *Un problema decisionale L è un problema la cui soluzione equivale ad una scelta tra 'sì' e 'no'.*

Definiamo ora le classi di complessità sopracitate.

Definizione 4.2.6 (Classe di complessità P). *Si definisce* classe P *l'insieme di tutti i problemi decisionali L per i quali esiste un algoritmo di risoluzione di costo polinomiale per ogni input z di L.*

La classe P comprende tutti quei problemi che possono essere ritenuti "facili", nel senso che esistono algoritmi di risoluzione efficienti nel senso della definizione 4.2.2.

La classe più importante ai fini della comprensione dei fondamenti dell'intrattabilità di un problema è la classe NP. Tale classe si definisce sulla base del concetto

di *algoritmo non deterministico*, dove il termine *non deterministico* si riferisce a quella modalità di calcolo che prevede la possibilità di scegliere l'operazione da eseguire entro un insieme definito, usufruendo del suggerimento di un agente esterno. Solitamente tale insieme di dati aggiuntivi forniti dall'esterno prende il nome di *certificato*.

In genere la risoluzione di un algoritmo non deterministico è limitata alle istanze di tipo 'sì'; si ha la seguente definizione.

Definizione 4.2.7 (Classe di complessità NP). *Si definisce* classe NP *l'insieme di tutti i problemi decisionali L per i quali esiste un algoritmo polinomiale in grado di verificare se L ammette soluzione di tipo 'sì' per ogni input z di L, sulla base di un opportuno insieme di dati aggiuntivi.*

Allo stesso modo è possibile definire la classe di problemi complementare a NP.

Definizione 4.2.8 (Classe di complessità co-NP). *Si definisce* classe co-NP *l'insieme di tutti i problemi decisionali L per i quali esiste un algoritmo polinomiale in grado di verificare se L ammette soluzione di tipo 'no' per ogni input z di L, sulla base di un opportuno insieme di dati aggiuntivi.*

Naturalmente tutti i problemi di classe P appartengono sia alla classe NP che a quella co-NP. Infatti la risoluzione polinomiale di un problema costituisce una verifica valida sia per le istanze di tipo 'sì' che per quelle di tipo 'no'.

Osservazione 4.2.1. Dalla definizione delle classi P e NP discende che

$$\mathrm{P} \subseteq \mathrm{NP};$$

attualmente non si è ancora riusciti a dimostrare la non validità dell'inclusione opposta. La validità dell'inclusione stretta

$$\mathrm{P} \subset \mathrm{NP}$$

è assunta come assioma.

Definizione 4.2.9 (Riducibilità). *Un problema decisionale L è detto riducibile polinomialmente (o Karp-riducibile) al problema L^* se esiste un algoritmo polinomiale A tale che:*

- *per ogni input z di L, A determina un input x di L^*;*
- *all'input z corrisponde la soluzione 'sì' di L se e solo se all'input x corrisponde la soluzione 'sì' di L^*.*

La classe dei problemi di NP più difficili da risolvere è detta *NP-complete*.

Definizione 4.2.10 (Classe di complessità NP-complete). *Un problema decisionale è detto* NP-completo *se ogni problema di NP è Karp-riducibile a L.*

In altri termini, un problema $L \in$ NP è NP-completo se ogni altro problema $L^* \in$ NP si può ridurre polinomialmente ad L, ossia ogni problema NP-completo è difficile almeno quanto tutti i problemi in NP.

Osservazione 4.2.2. Dalla definizione 4.2.10 discende che

$$\text{NP-complete} \subseteq \text{NP}.$$

La nozione di *completezza* è particolarmente importante per la classe NP. I problemi NP-completi sono verosimilmente *non-polinomiali*; vale infatti la seguente *proprietà caratterizzante dei problemi NP-completi:*

non esiste un algoritmo polinomiale $\forall L \in$ *NP-complete.*

Infatti, se esistesse un algoritmo polinomiale $\forall L \in$ NP-complete allora esisterebbe un algoritmo polinomiale $\forall L \in$ NP, quindi NP $\subseteq$ P; essendo P $\subseteq$ NP, si avrebbe P = NP, che è da escludere per l'osservazione 4.2.1.
Si tratta quindi di problemi la cui difficoltà è garantita.

La classe dei problemi NP-completi è non vuota. A tal proposito si consideri la seguente definizione:

Definizione 4.2.11 (Problema della 3-soddisfabilità). *Sia* $\{x_1, \ldots, x_n\}$ *un insieme di variabili booleane, ossia variabili il cui valore è "vero" o "falso". Sia* S *un'espressione booleana di 3 letterali per clausola, ossia*

$$S = (l_{11} + l_{12} + l_{13}) * (l_{21} + l_{22} + l_{23}) * \cdots * (l_{m1} + l_{m2} + l_{m3}),$$

dove ogni letterale l_{ij} *è uguale a* x_k *o alla sua negazione* $\bar{x}_k$ *e " +" e " ∗" indicano gli operatori logici "and" e " or" del calcolo proposizionale. Il problema della 3-soddisfabilità consiste nel verificare se l'espressione booleana* S *assume il valore "vero" per una particolare assegnazione delle variabili* x_i*.*

È possibile dimostrare il seguente fondamentale teorema (cfr. [34]).

Teorema 4.2.1 (Cook, 1971). *Il problema della 3-soddisfabilità è NP-completo.*

Osservazione 4.2.3. Una volta individuato un problema NP-completo A, per dimostrare l'appartenenza di un problema X a NP-complete non è necessario dimostrare la riducibiltà di ogni problema di NP ad X; è sufficiente verificare che $X \in$ NP e che X è riducibile ad A. Nonostante questa semplificazione, la parte più complicata è rappresentata dalla scelta della trasformazione che opera la riduzione.

I problemi computazionali più difficili da risolvere appartengono alla classe NP-hard:

Definizione 4.2.12 (Classe di complessità NP-hard). *Un problema decisionale è detto* NP-hard *se esiste almeno un problema NP-completo che sia Karp-riducibile a* L*.*

In altri termini, un problema L è NP-hard se ogni problema di NP si può ridurre polinomialmente a L (quindi L è difficile almeno quanto ogni problema di NP), ma non si è riusciti a dimostrare che $L \in$ NP; se questo accadesse, L sarebbe un problema NP-completo.

4.2.3 Problemi di ottimizzazione e problemi decisionali

Lo studio della complessità dei problemi di ottimizzazione si inquadra in quello dei problemi decisionali; è infatti possibile considerare una versione decisionale per ogni problema del tipo 1.1.1 (cfr. [63], capitolo 1 e capitolo 3 e [9]).
Innanzitutto si consideri la seguente:

Definizione 4.2.13 (Problema di ottimizzazione NP). *Un problema di ottimizzazione NP è costituito da:*

- *un insieme di istanze I che può essere individuato in un tempo polinomiale;*
- *un insieme di soluzioni ammissibili S per ogni istanza $i \in I$;*
- *una funzione obiettivo f che associa a ciascuna soluzione $s \in S$ un valore positivo $f(i, s)$, per ogni istanza $i \in I$;*
- *un obiettivo, che consiste nell'individuazione del minimo della f.*

L'associazione tra un problema decisionale e il corrispondente problema di ottimizzazione è definita come segue:

Definizione 4.2.14. *Sia dato un problema computazionale P consistente in un insieme di istanze I e un insieme di soluzioni ammissibili S; ad ogni istanza $i \in I$ corrisponda un insieme di soluzioni ammissibili $S_i \subseteq S$. Allora, dati $i \in I$ e il corrispondente insieme $S_i \subseteq S$, valgono le seguenti:*

- *la versione decisionale P_d di P consiste nello stabilire se $S_i = \emptyset$ oppure $S_i \neq \emptyset$;*
- *la versione di ottimizzazione P_o di P consiste nell'individuare un elemento di S_i.*

Quindi, per quanto affermato dalla definizione 4.2.14, risolvere un problema di ottimizzazione P_o equivale a risolvere un problema decisionale P_d.

Per concludere, osserviamo che un problema di ottimizzazione è considerato *trattabile* se di ottimizzazione locale (cfr. [74]) e *intrattabile* se di ottimizzazione globale (cfr. [49]).

4.3 Complessità dei problemi di ottimizzazione globale

Per lo studio della complessità del problema 1.1.1, poniamoci nel seguente contesto: se non diversamente precisato, il modello di riferimento sia la Macchina di Turing, la risorsa sia il tempo, l'efficienza si riferisca alla computazione polinomiale e lo studio della complessità si riferisca sempre al cosiddetto *caso peggiore*, ossia tra tutte le possibili istanze del problema si considerino quelle che richiedono un maggior numero di calcoli.

4.3.1 Programmazione quadratica

Consideriamo inizialmente alcuni risultati relativi a problemi di tipo quadratico che sono rilevanti per lo studio del problema generale 1.1.1.
Definiamo il problema di programmazione quadratica (QP) nel modo seguente.

$$\min \sum_{i \geq j} c_{ij} x_i x_j, \tag{4.1}$$

$$x \in S \equiv \{x \in [0,1]^n \mid Ax \leq b\}.$$

Supponiamo che gli algoritmi a cui ci si riferisce determinino una soluzione approssimata nel senso della definizione 2.2.7. Sia inoltre $\tilde{P}$ la classe dei problemi risolvibili in tempo quasi-polinomiale, ossia tali che il numero di operazioni richieste per risolvere un problema in $\tilde{P}$ sia limitato superiormente da $O(n^{(\log n)^k})$, dove k è una costante.
Valgono i seguenti teoremi (cfr. [28]).

Teorema 4.3.1. *Si supponga che NP $\not\subseteq \tilde{P}$. Allora non esiste un algoritmo polinomiale che determina una soluzione approssimata x di (4.1) secondo la definizione 2.2.7, con $\varepsilon = 1 - 2^{-(\log n)^\delta}$ e $\delta > 0$ costante opportuna.*

Poiché generalmente si ritiene che NP $\not\subseteq \tilde{P}$, non ci si può aspettare che un algoritmo risolva il problema (4.1) in tempo polinomiale.

Teorema 4.3.2. *Sia $P \neq NP$. Allora esiste una costante $\mu \in (0,1)$ per cui non esiste un algoritmo polinomiale che risolve il problema (4.1) secondo la definizione 2.2.7 con $\varepsilon = \mu$.*

In particolare, è possibile dimostrare che la costante μ può essere scelta inferiore a $\frac{1}{3}$.
Il teorema 4.3.2 asserisce che l'esistenza di un algoritmo polinomiale per la risoluzione del problema (4.1) implicherebbe $P = NP$, situazione esclusa nella teoria della complessità, come visto nell'osservazione 4.2.1.

Essendo il problema (4.1) un caso particolare del problema generale, vale il seguente risultato.

Proposizione 4.3.1. *Non è possibile individuare un'approssimazione del minimo globale del problema 1.1.1 in un tempo polinomiale, a meno che non sia $P = NP$.*

Per un qualsiasi algoritmo A considerato nel modello "black-box", vale il seguente teorema (Vavasis, 1995, [75]).

Teorema 4.3.3. *Sia $S = [0,1]^n$ e $F(k,p)$ la classe delle funzioni* k*-volte differenziabili in S e tali che la* k*-esima derivata sia limitata da p nel modo seguente:*

$$\left| \frac{d^k}{dt^k} f(x+tu) \right| \leq p,$$

per ogni $x \in S$ e per ogni vettore unitario u.
Sia A un algoritmo che opera nel modello "black-box" valutando sia f che le sue derivate. Supponiamo che, per $f \in F(k,p)$, A permetta di ottenere un punto x tale che $f(x) \leq f(x^*) + \varepsilon$, con x^* punto di minimo globale di f ed ε costante positiva prefissata.
Allora esiste una funzione $f \in F(k,p)$ tale che il numero di passi eseguiti dall'algoritmo A su f è almeno pari a $c\left(\frac{p}{\varepsilon}\right)^{\frac{n}{k}}$, con c costante positiva opportuna.

Questo teorema asserisce che, anche se le derivate sono limitate, la complessità di risoluzione del problema 1.1.1 cresce in modo esponenziale con la dimensione del problema. Ciò significa che il problema 1.1.1 non può essere risolto in modo efficiente.

Al paragrafo 2.3 si accennava al fatto che la disponibilità di informazioni di tipo globale sul problema non è in grado di garantire l'esistenza di algoritmi efficienti. Sulla base del teorema 4.3.3 si può considerare il seguente:

Esempio 4.3.1.
Si consideri una funzione lipschitziana f; $\forall x \in S$, è possibile conoscere una limitazione sul decremento della funzione in ogni intorno del punto considerato e, conseguentemente, ottenere un'approssimazione della soluzione senza dover necessariamente studiare insiemi di misura arbitrariamente piccola. In altre parole, supponiamo di voler individuare un'approssimazione del minimo globale di f secondo la definizione 2.2.3 con un'accuratezza ε. È possibile partizionare l'insieme ammissibile S in m sottoinsiemi in cui la massima variazione della funzione è inferiore ad ε; così facendo, sono sufficienti m valutazioni di funzione per determinare un'approssimazione dell'ottimo globale. D'altra parte però il teorema 4.3.3 asserisce che m cresce esponenzialmente al crescere della dimensione del problema, cosicché un simile algoritmo non è efficiente.

4.3.2 Pure Random Search

Si è già avuto modo di osservare al capitolo 3 che le proprietà di convergenza di algoritmi di tipo *deterministico*, ossia algoritmi che offrono assoluta garanzia

di successo, si possono migliorare introducendo elementi di tipo probabilistico, come ad esempio le *scelte casuali*. Algoritmi di questo tipo vengono appunto detti *stocastici*. Il più semplice tra gli algoritmi stocastici per l'ottimizzazione globale è il *Pure Random Search*, descritto al paragrafo 3.2.

Dal punto di vista della complessità computazionale, è possibile dimostrare che il costo in termini di valutazioni di funzione richiesto dal PRS per determinare un punto di minimo globale di (1.1) cresce enormemente quando la dimensione di S tende all'infinito; si può dimostrare infatti il seguente lemma (cfr. [28]):

Lemma 4.3.1. *Sia S un insieme limitato di dimensione n. Allora il numero medio di valutazioni di funzione $E(n)$ richiesto dall'applicazione dell'algoritmo PRS per individuare per la prima volta un punto $x_k \in S^{**}$ è*

$$E(n) \simeq \pi\sqrt{2e}\left(\frac{n}{2\pi e}\right)^{\frac{n+1}{2}}\left(\frac{1}{\varepsilon}\right)^n mis(S),$$

con n sufficientemente grande, essendo

$$S^{**} = \{x \in S : \ \|x - x^*\|_2 \leq \varepsilon, \ x^* \text{ punto di minimo globale di } f\},$$

e *il numero di Nepero e $mis(S)$ la misura di S.*

La dimostrazione di questo lemma si basa sulle formule per il calcolo del volume di un'ipersfera di $\mathbb{R}^n$.
Un risultato formalmente più semplice si ottiene considerando la norma l_∞. Ricordiamo che

$$x \equiv (x_i) \in l_\infty \text{ se } \ \|x\|_\infty = \sup_i |x_i| < +\infty.$$

In tal caso il risultato del lemma 4.3.1 assume la forma:

$$E(n) \simeq \left(\frac{1}{\varepsilon}\right)^n mis(S).$$

4.3.3 Pure Adaptive Search

Risultati molto interessanti sono stati ottenuti da Zabinsky e Smith in [77] relativamente all'algoritmo concettuale *Pure Adaptive Search*, basato anch'esso sulla generazione casuale di punti nello spazio di ricerca (si veda il paragrafo 3.4.1).

Per quanto riguarda la complessità dell'algoritmo PAS, è stato dimostrato il seguente risultato:

Teorema 4.3.4. *Si consideri il problema di minimizzazione globale 1.1.1 su un insieme convesso S di dimensione n con diametro al più d, e con costante di Lipschitz al più L. Allora, per ogni y tale che*

$$y_* \leq y \leq y^*,$$

con $y_* = \min_{x \in S} f(x)$ *e* $y^* = \max_{x \in S} f(x)$, *il numero atteso di iterazioni* $E(N^*)$ *necessarie per ottenere un valore inferiore o uguale ad* y *soddisfa la seguente limitazione:*

$$E(N^*) \leq 1 + \left(\ln \left(\frac{Ld}{y - y_*} \right) \right) n.$$

Dato che il PAS esegue una valutazione di funzione ad ogni iterazione, dal teorema 4.3.4 si evince che esiste una limitazione superiore per il numero atteso di valutazioni di funzione necessarie ad individuare l'ottimo globale, limitazione che dipende linearmente dalla dimensione del problema.
Nonostante i risultati interessanti dal punto di vista computazionale, si è già osservato come nella pratica si presentino grosse difficoltà nell'effettiva applicazione del PAS. Tale algoritmo è comunque molto interessante come punto di partenza per la costruzione di algoritmi effettivamente implementabili.

4.3.4 Adaptive Search

L'algoritmo Adaptive Search introdotto al paragrafo 3.4.2 rappresenta una generalizzazione del PAS e ne conserva il risultato di linearità. È stato infatti dimostrato da Romeijn e Smith (1994, [59]) il teorema seguente:

Teorema 4.3.5. *Si consideri il problema di minimizzazione globale 1.1.1 su un insieme convesso* S *di dimensione* n *con diametro al più* d, *e con costante di Lipschitz al più* L. *Allora, per ogni* y *tale che*

$$y_* \leq y \leq y^*,$$

con $y_* = \min_{x \in S} f(x)$ *e* $y^* = \max_{x \in S} f(x)$, *il numero atteso di iterazioni* $E(N^{AS})$ *necessarie per ottenere un valore inferiore o uguale ad* y *mediante l'algoritmo AS soddisfa la seguente limitazione:*

$$E(N^{AS}) \leq E(N^{PAS}) \leq 1 + \left(\ln \left(\frac{Ld}{y - y_*} \right) \right) n,$$

essendo $E(N^{PAS})$ *il numero atteso di iterazioni richieste dal PAS.*

Da questo teorema si può dedurre che se è possibile trovare un modo per generare punti in accordo con la distribuzione di Boltzmann π_T per qualche $T > 0$, allora si può risolvere in modo efficiente un'ampia classe di problemi del tipo 1.1.1.

4.3.5 Global Minimization

In [27], Gaviano e Lera hanno studiato la complessità di un algoritmo generale di minimizzazione globale (*Global Minimization*) che può essere inquadrato nella classe dei metodi "Multistart". Tale algoritmo applica ad ogni punto generato a caso su S una procedura di ricerca locale, l'*algoritmo di più ripida discesa* combinato

con la procedura di Armijo per la determinazione del passo. La procedura di ricerca locale mira ad individuare il minimo locale della regione di attrazione alla quale il punto appartiene ossia, in generale, punti nell'insieme:

$$G_\varepsilon = \{x \in S : \ \|\nabla f(x)\| \leq \varepsilon\}.$$

Ogni nuovo punto viene accettato solo se il valore corrispondente della f è inferiore a quello nel punto corrente. La soluzione viene approssimata secondo la definizione 2.2.3, ossia: dato $\eta > 0$, si vuole determinare un punto in S^{**}, essendo qui

$$S^{**} \equiv \{x \in S : \ |f(x) - f(x^*)| \leq \eta, \ x^* \text{ punto di minimo globale di } f\}.$$

Lo schema generale dell'algoritmo è il seguente:

Algoritmo 4.3.1 (Global Minimization, GM).

sia $k \leftarrow 0$;
si scelga y con un campionamento uniforme su S;
$x_0 \leftarrow local\_search(y)$;
$fx_0 \leftarrow f(x_0)$;
loop
  si scelga y con un campionamento uniforme su S;
  $y' \leftarrow local\_search(y)$;
  $fy' \leftarrow f(y')$;
  if $f(y') < fx_k$
    $x_{k+1} \leftarrow y'$;
    $fx_{k+1} \leftarrow f(y')$;
  else
    $x_{k+1} \leftarrow x_k$;
    $fx_{k+1} \leftarrow fx_k$;
  endif
  $k \leftarrow k + 1$;
forever
end

La procedura di ricerca locale (*local_search*) è realizzata dall'algoritmo:

Algoritmo 4.3.2 (Local Search, LS).

sia $x_0 \in \mathbb{R}^n$ e $i \leftarrow 0$;
repeat
  si scelga $\lambda_i > 0$ tale che $f(x_i - \lambda_i \nabla f(x_i)) \leq f(x_i)$;
  $x_{i+1} = x_i - \lambda_i \nabla f(x_i)$;
  $i \leftarrow i + 1$;
until $\|\nabla f(x_i)\| \leq \varepsilon$.
end

Poiché la procedura di ricerca locale LS individua punti in G_ε, si suppone che ε abbia un valore sufficientemente piccolo perché risulti che $x \in S^{**},\ \forall x \in G_\varepsilon$.

La scelta del passo λ_i in LS è effettuata secondo la *procedura di Armijo*; in particolare, dati un punto $x_i \in S$ e una direzione di ricerca h_i tale che $-h_i \nabla f(x_i) > 0$, si ha:

Algoritmo 4.3.3 (Procedura di scelta di Armijo).

siano $\alpha, \beta \in (0,1)$, $\rho > 0$, $h_i, x_i \in S$;
sia:

$$\theta(\lambda, x_i) = f(x_i + \lambda h_i) - f(x_i) - \lambda \alpha h_i^T \nabla f(x_i);$$

$\mu \leftarrow \rho$;
loop
  si calcoli $\theta(\mu, x_i)$;
  if $\theta(\mu, x_i) < 0$
    $\lambda_i \leftarrow \mu$;
    stop;
  endif
  $\mu \leftarrow \beta\mu$;
forever
end

Perché la scelta del passo λ secondo la procedura di Armijo sia ben definita, occorre formulare alcune ipotesi:

1. l'insieme S sia compatto e $A = \{x \in S :\ \|\nabla f(x)\| = 0\}$ sia un insieme finito;
2. $\forall x \in S$ tale che $\nabla f(x) \neq 0$, esista $\lambda > 0$ per cui

$$\nabla f(x - \tilde{\lambda} \nabla f(x))^T \nabla f(x) = 0,$$

dove $x - \tilde{\lambda} \nabla f(x)$ appartiene all'interno di S.

Risultati locali

In generale è noto che il problema dell'individuazione di un minimo locale di una funzione è *trattabile*, ossia può essere risolto mediante algoritmi che richiedono un numero polinomiale di valutazioni di funzione e di gradiente (si vedano ad esempio i lavori di Vavasis [73] e [74]).

In [27] gli autori dimostrano alcuni risultati relativi sulla complessità della procedura LS del tutto simili a quelli di Vavasis in [74]. Tali risultati sono alla base della successiva trattazione globale del problema 1.1.1.

È possibile verificare il seguente lemma:

Lemma 4.3.2. *Sia* $\bar{x} = x - \bar{\lambda}\nabla f(x)$, *con* $x \in S \setminus G_\varepsilon$ *e sia* $\bar{\lambda}$ *il passo calcolato secondo la procedura di Armijo. Allora*

$$f(\bar{x}) - f(x) \leq -\beta^{j^*}\alpha\rho\varepsilon^2,$$

con j^* *il più piccolo intero positivo tale che*

$$\beta^{j^*} < \frac{2(1-\alpha)}{M\rho},$$

essendo $M = \max_{x\in S} \|H(x)\|_2$ *e* $H(x)$ *l'hessiana di* f *in* x.

Dal lemma 4.3.2 discende il seguente:

Teorema 4.3.6. *Il numero di valutazioni della funzione e del gradiente richieste dalla ricerca locale LS con la procedura di Armijo per soddisfare il criterio di stop* $\|\nabla f(x_i)\| \leq \varepsilon$, *sono al massimo:*

$$(j^*+1)\frac{(\overline{f} - \underline{f})}{\beta^{j^*}\alpha\rho\varepsilon^2} \quad e \quad \frac{(\overline{f} - \underline{f})}{\beta^{j^*}\alpha\rho\varepsilon^2}$$

rispettivamente, essendo $\overline{f} = \max_{x\in S} f(x)$ *e* $\underline{f} = \min_{\|\nabla f(x)\|\geq\varepsilon} f(x)$.

Essendo dimostrabile (cfr. [28]) che:

$$(\overline{f} - \underline{f}) \leq 2Mn,$$

dal teorema 4.3.6 discende che il numero di valutazioni di f e del suo gradiente sono limitati superiormente da

$$(j^*+1)\frac{2Mn}{\beta^{j^*}\alpha\rho\varepsilon^2} \quad \text{e} \quad \frac{2Mn}{\beta^{j^*}\alpha\rho\varepsilon^2}$$

rispettivamente.

Questo significa che l'individuazione di una soluzione locale approssimata di (1.1) mediante la procedura LS richiede un numero di valutazioni di funzione e di gradiente che cresce linearmente con la dimensione del problema. Se dal punto di vista teorico il costo di LS è lo stesso dell'algoritmo proposto da Vavasis in [74], dal punto di vista pratico gli autori si attendono da LS una maggior efficienza.

Risultati globali

Per l'algoritmo GM applicato alla minimizzazione globale si possono dimostrare i seguenti risultati (cfr. [27]).

Lemma 4.3.3. *Sia A^* l'insieme definito nel modo seguente:*

$$\begin{aligned} A^* = \{&x \in S \text{ tale che la procedura di ricerca locale applicata ad } x, \\ &\text{con criterio di stop } \|\nabla f(x_i)\| \leq \varepsilon \\ &\text{e procedura di Armijo per la scelta del passo,} \\ &\text{individua un punto in } S^{**}\}; \end{aligned}$$

allora A^* è misurabile.

L'insieme A^* non è altro che la regione di attrazione di un generico punto di minimo locale.

Teorema 4.3.7. *Si supponga che il costo computazionale di una valutazione del gradiente della funzione f equivalga a n valutazioni di f. Allora, il numero atteso di valutazioni di funzione richiesto dall'algoritmo GM, combinato con il metodo di più ripida discesa e la procedura di Armijo, è inferiore o pari a*

$$\frac{1}{p^*}\left(t^*(1+n+j^*)\right),$$

dove:

- $t^* = \frac{\overline{f}-\underline{f}}{\alpha\rho\varepsilon^2\beta^{j^*}}$;
- j^* *è il più piccolo intero positivo tale che* $\beta^{j^*} < \frac{2(1-\alpha)}{M\rho}$;
- $p^* = mis(A^*)/mis(S)$.

Si osservi che p^* rappresenta la probabilità che la ricerca locale individui un punto in S^{**} partendo da un qualunque punto $x \in S$.

Da un confronto tra i risultati dei teoremi 4.3.2 e 4.3.3 con quello fornito dal teorema 4.3.7 emerge che ad una complessità che è funzione esponenziale della dimensione, come mostrato dai primi due teoremi per il *caso peggiore*, si contrappone, seppur sotto opportune ipotesi, una complessità di tipo lineare in n, quando si sfrutta la misura delle regioni di attrazione dei minimi locali.
Questo significa che problemi di ottimizzazione di dimensione via via crescente, in cui la misura delle regioni di attrazione non decresce, hanno una complessità di tipo lineare.
Tuttavia esistono svariati problemi reali per cui il numero dei minimi aumenta vertiginosamente con la dimensione del problema, con una conseguente riduzione della misura delle regioni di attrazione. In tal caso la complessità è di tipo esponenziale.

Capitolo 5

Un nuovo metodo per l'ottimizzazione globale

5.1 Introduzione

Per la ricerca di un minimo locale del problema 1.1.1 sono stati progettati numerosi algoritmi, sia generali che specifici per situazioni particolari quali, ad esempio, problemi di piccole o grandi dimensioni o problemi non differenziabili. La maggior parte di questi metodi risultano molto efficienti nella loro applicazione. Normalmente essi generano, a partire da un punto iniziale x_0, una successione di punti x_i in cui la funzione è decrescente e converge ad un minimo locale. Il punto x_0 è generalmente arbitrario ed il minimo locale trovato f^* risulta essere il minimo globale della regione di attrazione di x^*, $Attr(f(x^*))$.
Sostanzialmente si individua il minimo globale $f^* = f(x^*)$ di un sottoinsieme $Attr(f^*)$ dell'insieme ammissibile, senza che venga esplorato tutto l'insieme dei punti a partire dai quali l'algoritmo individua f^*. Sfortunatamente in generale non siamo in grado di conoscere e valutare la misura di questo sottoinsieme. A tale proposito sono significativi i risultati di Gaviano e Lera (cfr. [27]) e Zabinski (cfr. [77]), enunciati nel capitolo 4. Nel primo lavoro si dimostra che la complessità computazionale è legata all'ampiezza della regione di attrazione. Nel secondo si dimostra che se si potesse individuare la regione $L_i(f) = \{x \in S \mid f(x) \leq f(x_i)\}$ ed operare su di essa, si potrebbe avere un algoritmo che trova il minimo globale con complessità lineare. Un algoritmo che utilizzando la conoscenza della costante di Lipschitz approssima con una successione di intervalli l'insieme $L_i(f)$, è stato recentemente proposto da Gaviano e Lera (cfr. [24]). L'algoritmo risulta utilizza-

bile per problemi di piccola dimensione.
Algoritmi che sfruttano la ricerca locale sono stati utilizzati per la minimizzazione globale, come ad esempio il Multistart o il Simulated Annealing con ricerca locale (si veda a tal proposito il capitolo 3). I principali problemi che si devono superare nell'applicazione di questi algoritmi sono i seguenti:

- evitare di ritrovare gli stessi minimi;
- disporre di un efficiente criterio di stop.

Il metodo *Simulated Annealing* supera il primo problema permettendo che si accettino dei punti in cui il valore della funzione può crescere secondo una opportuna distribuzione di probabilità. L'algoritmo proposto da Abaffy et al. in [2] risolve lo stesso problema applicando una ricerca locale solo se il punto iniziale x_0 della nuova ricerca locale è tale che $f(x_0)$ sia minore dell'ultimo minimo locale trovato. In questo modo si effettuano solo le ricerche locali strettamente necessarie a trovare il minimo globale, ma allo stesso tempo si riduce la probabilità di trovarlo. La situazione è illustrata nella figura 5.1: risulta evidente che la probabilità di trovare il punto di minimo globale l_2 dopo aver trovato l_1 è notevolmente ridotta se si accettano come punti iniziali solo punti in cui il valore della funzione è minore di $f(l_1)$.
In questo contesto si inserisce l'algoritmo che verrà ora presentato.

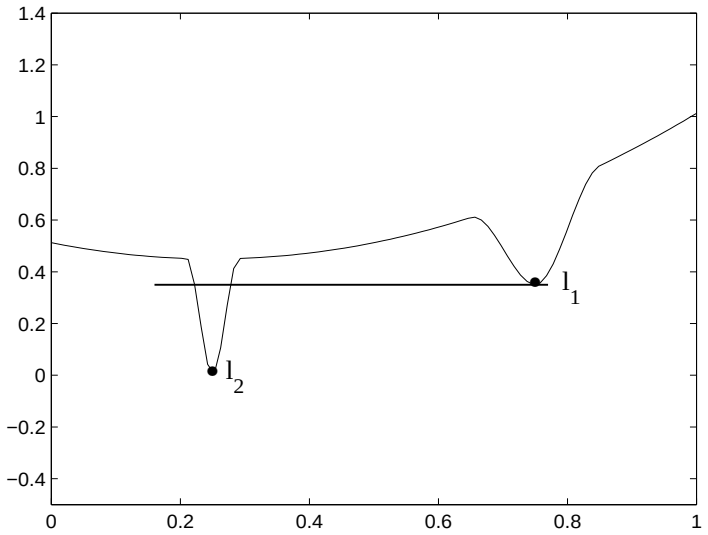

Figura 5.1: Situazione relativa ad un passo dell'algoritmo di Abaffy et al.

5.2 Proprietà teoriche

L'idea dell'algoritmo che si vuole introdurre in questo capitolo, consiste nello scegliere ad ogni iterazione un punto x_0 uniformemente su S e nell'applicare sempre una procedura di ricerca locale a partire da esso nel caso in cui $f(x_0)$ è minore del valore fl_i dell'ultimo minimo locale trovato, mentre si applica la ricerca locale con probabilità d_i se $f(x_0)$ risulta maggiore di fl_i. I valori d_i saranno calcolati in modo che il numero medio di valutazioni di funzione necessarie per ottenere un nuovo ottimo locale sia minimo. Lo schema generale dell'algoritmo è il seguente:

Algoritmo 5.2.1 (Algoritmo Glob).

si scelga x_0 uniformemente in S;
$i \leftarrow 1;\ \ j \leftarrow 1;$
$(x_1, fx_1) \leftarrow local\_search(x_0);$
$l_i = x_1; fl_i = fx_1;$
repeat
  $j \leftarrow j + 1;$
  si scelga x_0 uniformemente in S;
  if $f(x_0) \leq fl_i$ oppure $(f(x_0) > fl_i$ e $rand(1) < d_i)$
    $(x_1, fx_1) \leftarrow local\_search(x_0);$
    if $fx_1 < fl_i$
      $l_i \leftarrow x_1;\ \ fl_i \leftarrow fx_1;$
      $i \leftarrow i + 1;$
    end if
  end if
until è soddisfatto un criterio di stop;
end

Nell'algoritmo *Glob* la procedura $local\_search(\cdot)$ indica un algoritmo di minimizzazione che trova un minimo locale a partire da un punto iniziale assegnato; $rand(1)$ indica un generatore di un numero a caso nell'intervallo $[0, 1]$.

Si può dimostrare il seguente teorema.

Teorema 5.2.1. *La funzione $f(\cdot)$ abbia un numero finito di minimi locali. Si consideri un'applicazione dell'algoritmo Glob; allora la probabilità che fl_i sia il minimo globale tende a uno per $j \to \infty$.*

Dimostrazione. Il caso in cui ogni minimo locale è globale è banale. Pertanto si supponga che $f(\cdot)$ abbia uno o più minimi locali che non sono globali. Sia fl_k il valore della funzione trovato che non dà la soluzione globale. Allora esiste un sottoinsieme G di S, di misura maggiore di zero, tale che $f(\cdot)$ assume un valore minore di fl_k, $\forall x \in G$. La probabilità di non trovare il minimo globale ad una generica iterazione dell'algoritmo *Glob* è minore di uno. Poiché la probabilità di

non trovare il minimo globale al crescere del numero delle iterazioni tende a zero, ne consegue il teorema. □

Nelle pagine seguenti si enunciano le motivazioni teoriche che permetteranno di valutare i valori d_i nell'algoritmo *Glob*. Si considerano due casi:

a) l'algoritmo *Glob* viene eseguito per un numero infinito di iterazioni;

b) l'algoritmo *Glob* viene eseguito per n iterazioni.

Per il problema 1.1.1 si considerano le seguenti ipotesi:

Ipotesi 5.2.1.

i) $f(\cdot)$ ha m punti di minimo locale $l_i,\ i = 1, ..., m$ e $f(l_i) > f(l_{i+1})$;

ii) $mis(S) = 1$.

Inoltre si considerino le seguenti definizioni per la funzione $f(\cdot)$ e l'algoritmo *Glob*.

Definizione 5.2.1.

- $A_{0,j} \equiv \{x_0 \in S \mid$ *partendo da* x_0, $local\_search(\cdot)$ *converge al punto di minimo* $l_j\}$;
- $A_{i,j} \equiv \{x_0 \in S \mid f(x_0) \leq f(l_i)$; *partendo da* x_0, $local\_search(\cdot)$ *converge al punto di minimo* $l_j\}$;
- $p_{0,j} = mis(A_{0,j})$;
- $p_{i,j} = mis(A_{i,j})$;
- $t_i \equiv$ *probabilità che avendo trovato il punto di minimo locale* l_i, *in una iterazione successiva non si trovi un nuovo minimo locale;*
- $(i_1, ..., i_p) \equiv$ *insieme (traiettoria) di p punti di minimo locale* $l_{i_1}, ..., l_{i_p}$ *trovati in una applicazione dell'algoritmo Glob;*
- $Prob_{i,j} \equiv$ *probabilità che l'algoritmo passi dal punto di minimo locale* l_i *al punto di minimo locale* l_j *in una generica iterazione;*
- $Prob_{i,j}^{(\infty)} \equiv$ *probabilità che l'algoritmo Glob passi da* l_i *a* l_j *supponendo che esso possa iterare un numero infinito di volte;*
- $Prob_t^{(n)} \equiv$ *probabilità che l'algoritmo Glob generi la generica traiettoria* $t = (i_1, ..., i_p)$ *in n iterazioni.*

Per l'ipotesi fatta risulta:

$$\sum_{j=1}^{m} p_{0,j} = mis(S) = 1.$$

Caso a) numero infinito di iterazioni

Si calcola ora il numero medio di valutazioni di funzione affinché l'algoritmo *Glob* trovi un minimo globale, supponendo di conoscere i valori $p_{0,j}$ e $p_{i,j}$, $i = 1, ..., m-1$ e $j = 1, ..., m$. Inoltre nel seguito si assumerà che il numero di valutazioni di funzione richiesto da *local_search* sia $k =$ costante.

Si dimostrano innanzitutto due lemmi.

Lemma 5.2.1.

$$\begin{aligned} t_i &= \sum_{j=1}^{i} p_{0,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) (1 - d_i), \\ Prob_{i,j} &= p_{i,j} + d_i(p_{0,j} - p_{i,j}), \\ Prob_t^{(n)} &= p_{0,i_1} \cdot (p_{i_1,i_2} + (p_{0,i_2} - p_{i_1,i_2})d_{i_1}) \cdot \ldots \cdot \\ &\cdot \quad (p_{i_{p-i},i_p} + (p_{0,i_p} - p_{i_{p-1},i_p})d_{i_{p-1}}) \cdot \\ &\cdot \sum_{\substack{j_1,\ldots,j_{p-1}=0 \\ j_1+\ldots+j_{p-1} \leq n-p}}^{n-p} t_{i_1}^{j_1} \cdot t_{i_2}^{j_2} \cdot \ldots \cdot t_{i_{p-1}}^{j_{p-1}}. \end{aligned}$$

Dimostrazione. La prima affermazione del lemma si ottiene osservando che se x_0 è scelto in $A_{i,j}$ per $j = 1, ..., i$, allora sicuramente non viene trovato un nuovo minimo locale; mentre se x_0 viene scelto in $A_{0,j} - A_{i,j}, j = i+1, ..., m$, allora si resta in l_i con probabilità $(1 - d_i)$.
La seconda affermazione segue dal fatto che si ottiene un nuovo minimo locale sia nel caso in cui il punto di partenza x_0 venga scelto in $A_{i,j}$ sia, con probabilità d_i, nel caso di scelta in $A_{0,j} - A_{i,j}$.
Si può verificare che $Prob_{i,j}$ costituisce una distribuzione di probabilità quando $j = i+1, ..., m$; infatti:

$$\sum_{l=i+1}^{m} (p_{i,l} + d_i(p_{0,l} - p_{i,l})) + \alpha = 1,$$

in cui α è la probabilità di restare in l_i, ossia:

$$\alpha = \sum_{l=1}^{i} p_{0,l} + \sum_{l=i+1}^{m} (1 - d_i)(p_{0,l} - p_{i,l}).$$

La terza affermazione è ottenuta osservando che, per ottenere la generica traiettoria $t = (i_1, ..., i_p)$, occorre prima ottenere il minimo in l_{i_1}, e questo si verifica con probabilità p_{0,i_1}; successivamente si deve passare dal minimo in l_{i_1} al minimo in l_{i_2} e via dicendo, fino ad ottenere il minimo in l_{i_p}. Allo stesso tempo può accadere che si resti j_1 volte in l_{i_1}, j_2 volte in l_{i_2}, e così via. Il lemma è così dimostrato. □

Lemma 5.2.2.

$$Prob_{i,j}^{(\infty)}(d_i) = \frac{p_{i,j} + d_i(p_{0,j} - p_{i,j})}{\sum_{l=i+1}^{m}(p_{i,l} + d_i(p_{0,l} - p_{i,l}))}, \tag{5.1}$$

$$Prob_{(i_1,...,i_p)}^{(\infty)}(d_{i_1},...,d_{i_{p-1}}), = p_{0,i_1} \cdot Prob_{i_1,i_2}^{(\infty)} \cdot ...Prob_{i_{p-1},i_p}^{(\infty)}, \tag{5.2}$$

con $i_p = i_m$.

Dimostrazione. La prima affermazione del lemma segue dal fatto che applicando l'algoritmo per n iterazioni, la probabilità di trovare l_j è data da:

$$(p_{i,j} + d_i(p_{0,j} - p_{i,j})) \cdot \sum_{l=0}^{n-1} \alpha^l, \tag{5.3}$$

quindi, per $n \to \infty$, si ottiene:

$$\frac{p_{i,j} + d_i(p_{0,j} - p_{i,j})}{1 - \alpha}, \tag{5.4}$$

cioè la (5.1).
Per quanto riguarda la (5.2), si osservi innanzitutto che ogni traiettoria per cui $i_p \neq i_m$ ha probabilità zero di essere generata. Nel caso in cui $i_p = i_m$, la probabilità $Prob_t^{(n)}$ della traiettoria t eseguendo solo n iterazioni è data nel lemma 5.2.1. Supponendo per semplicità che $n - p$ sia un multiplo di p, abbiamo:

$$\begin{aligned} & \sum_{j_1=0}^{(n-p)/p} t_{i_1}^{j_1} \cdot ... \cdot \sum_{j_{p-1}=0}^{(n-p)/p} t_{i_{p-1}}^{j_{p-1}} \\ \leq & \sum_{\substack{j_1,...,j_{p-1}=0 \\ j_1+...+j_{p-1} \leq n-p}}^{n-p} t_{i_1}^{j_1} \cdot t_{i_2}^{j_2} \cdot ... \cdot t_{i_{p-1}}^{j_{p-1}} \\ \leq & \sum_{j_1=0}^{n-p} t_{i_1}^{j_1} \cdot ... \cdot \sum_{j_{p-1}=0}^{n-p} t_{i_{p-1}}^{j_{p-1}}. \end{aligned} \tag{5.5}$$

Poiché per $n \to \infty$ si ottengono le seguenti:

$$\sum_{j_1=0}^{\infty} t_{i_1}^{j_1} = \frac{1}{1 - t_{i_1}}, ..., \sum_{j_{p-1}=0}^{\infty} t_{i_{p-1}}^{j_{p-1}} = \frac{1}{1 - t_{i_{p-1}}}, \tag{5.6}$$

con il passaggio al limite in $Prob_t^{(n)}$ si ricava la (5.2). □

Si ha infine il seguente teorema:

Teorema 5.2.2. *Il numero medio fun_ev di valutazioni di funzione affinché l'algoritmo Glob trovi un punto di minimo globale è dato da:*

$$fun\_ev(d_1, ..., d_m) = \sum_{t \in T} Prob_t^{(\infty)} \cdot (k + f_{i_1,i_2}, ... + f_{i_{p-1},i_p}), \tag{5.7}$$

in cui T indica l'insieme di tutte le traiettorie possibili in cui l'ultimo minimo locale è il minimo globale.

Dimostrazione. Il numero di valutazioni di funzione per avere il primo minimo locale è:

$$f_0 = k. \tag{5.8}$$

Il numero medio f_i di valutazioni di funzione per una singola scelta di x_0 (ossia per ogni iterazione di *Glob*) stando nel punto di minimo l_i, è dato da:

$$f_i = k \sum_{l=i+1}^{m} p_{i,l} + kd_i \left(1 - \sum_{l=i+1}^{m} p_{i,l}\right) + (1 - d_i) \left(1 - \sum_{l=i+1}^{m} p_{i,l}\right). \tag{5.9}$$

Il numero medio di valutazioni di funzione $f_{0,j},\ j = 1, ..., m$, per ottenere l_j alla prima iterazione è dato da:

$$f_{0,j} = k,\ j = 1, ..., m. \tag{5.10}$$

Il numero medio di valutazioni di funzione $f_{i,j}$ per andare da l_i a l_j è:

$$f_{i,j} = f_i \frac{1}{Prob_{i,j}},\quad i = 1, ..., m-1,\ j = i+1, ..., m. \tag{5.11}$$

Tenuto conto dell'insieme delle traiettorie possibili e delle loro probabilità, si ottiene il teorema. □

Poniamoci dunque il seguente problema:

Problema 5.2.1. *Sia dato il problema 1.1.1 e siano noti i valori k, $p_{0,j}$ e $p_{i,j}$. Determinare i valori $d_i^*,\ i = 1, ..., m$, tali che:*

$$fun\_eval(d_1^*, ..., d_m^*) = \min_{d_1,...,d_m} fun\_eval(d_1, ..., d_m).$$

Esempio 5.2.1.
Consideriamo il caso $m = 3$. In tal caso si ha possono avere 4 traiettorie e l'insieme T è definito come:

$$T = \{(3), (1, 2, 3), (1, 3), (2, 3)\}. \tag{5.12}$$

Dall'equazione (5.7) si trova per fun_ev :

$$
\begin{aligned}
& p_{0,3} \cdot k + \\
+ \quad & p_{0,1} \frac{p_{1,2} + d_1(p_{0,2} - p_{1,2})}{p_{1,2} + p_{1,3} + d_1(p_{0,2} - p_{1,2} + p_{0,3} - p_{1,3})} \cdot \\
\cdot \quad & \left(k + \frac{f_1}{p_{1,2} + d_1(p_{0,2} - p_{1,2})} + \frac{f_2}{p_{2,3} + d_2(p_{0,3} - p_{2,3})} \right) + \\
+ \quad & p_{0,1} \frac{p_{1,3} + d_1(p_{0,3} - p_{1,3})}{p_{1,2} + p_{1,3} + d_1(p_{0,2} - p_{1,2} + p_{0,3} - p_{1,3})} \cdot \\
\cdot \quad & \left(k + \frac{f_1}{p_{1,3} + d_1(p_{0,3} - p_{1,3})} \right) + \\
+ \quad & p_{0,2} \left(k + \frac{f_2}{p_{2,3} + d_2(p_{0,3} - p_{2,3})} \right).
\end{aligned} \tag{5.13}
$$

I grafici nella figura 5.2 illustrano l'andamento di fun_ev al variare di d_1 e d_2 nell'intervallo $[0, 1]$. I valori $p_{i,j}$ sono dati nella tabella 5.1. Nel primo grafico si è posto $k = 16$, nel secondo k=8.

Dall'analisi dei grafici risulta che il minimo è raggiunto nel primo caso per $d_1 = 0$ e $d_2 = 0$; nel secondo caso per $d_1 = 0$ e $d_2 = 1$. Ciò suggerisce che il minimo debba trovarsi solo in corrispondenza dei vertici del cubo $[0, 1]^n$ in $\mathbb{R}^n$; ovvero che la probabilità di effettuare una ricerca locale debba essere zero oppure uno. Resta da verificare che tale proprietà vale in generale.

$p_{0,1} = 0.5$	$p_{0,2} = 0.4$	$p_{0,3} = 0.1$
$p_{1,2} = 0.05$	$p_{1,3} = 0.04$	$p_{2,3} = 0.01$

Tabella 5.1: Valori delle probabilità per il calcolo di fun_eval

Esempio 5.2.2.
Nella figura 5.3 si riportano i grafici del numero delle valutazioni di funzione al variare di d_i nell'intervallo $[0, 1]$ per i quattro minimi della funzione test di Goldstein e Price (si veda l'Appendice); il numero di valutazioni di funzione viene calcolato sia usando l'equazione (5.7) e sia sperimentalmente con 1000 esecuzioni dell'algoritmo *Glob*, ciascuna con 1000 iterazioni. I valori p_{ij} utilizzati in fun_ev sono calcolati ugualmente sperimentalmente. Il numero di valutazioni di funzione k richiesto dalla ricerca locale è indicato nella figura con $k\ tot$, mentre *limite* indica un valore di riferimento con cui confrontare $k\ tot$, definito successivamente in (5.18). I grafici mostrano una buona concordanza tra i dati sperimentali e quelli calcolati teoricamente.

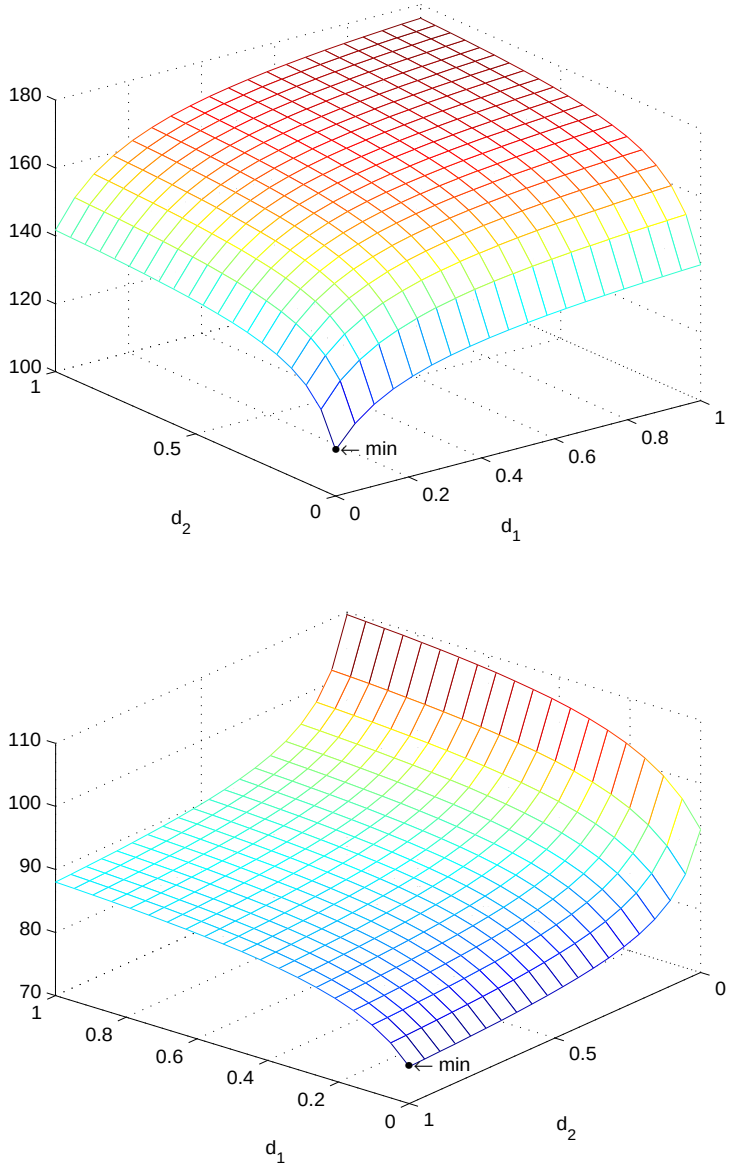

Figura 5.2: Numero medio di valutazioni di funzione con k=16 e k=8

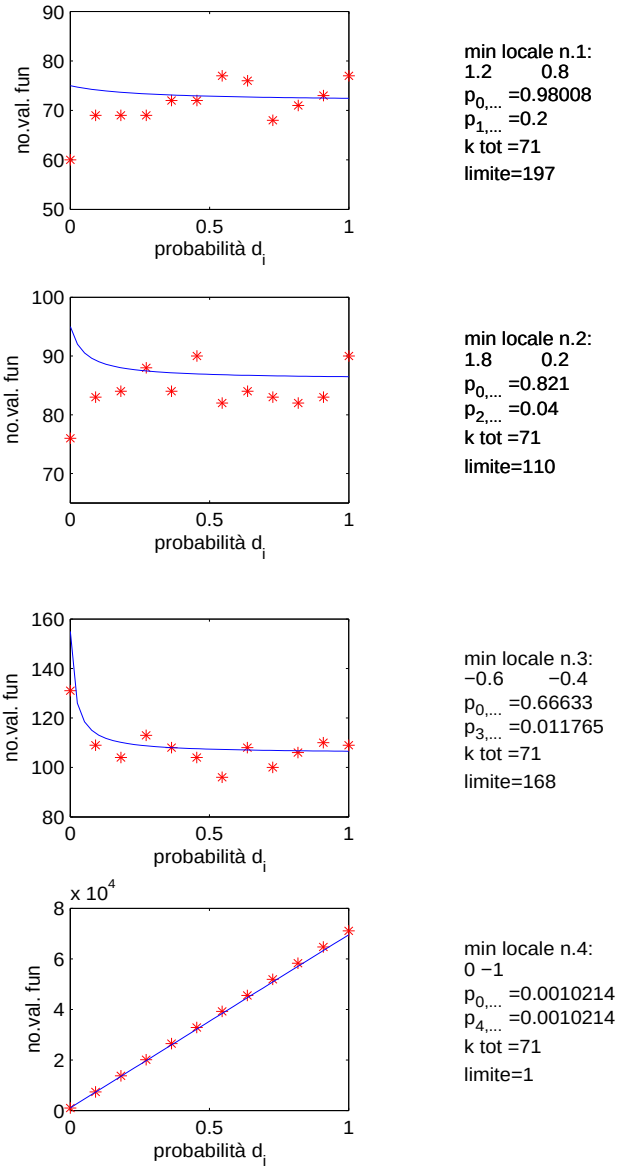

Figura 5.3: Confronto tra il numero medio di valutazioni di funzione per la funzione test di Goldstein e Price, calcolato teoricamente (linea continua) e sperimentalmente

Analisi Locale

Essendo l'equazione (5.7) difficile da studiare, procediamo con un'analisi locale dell'algoritmo, cioè non consideriamo tutti i passaggi dell'algoritmo attraverso tutti i minimi locali, ma analizziamo solo il passaggio da un qualunque minimo ad un altro. Si calcola pertanto il numero medio di valutazioni di funzione affinché l'algoritmo *Glob*, avendo trovato un minimo locale $f(l_i)$, trovi un nuovo minimo locale (qualsiasi).

Si può dimostrare il seguente teorema.

Teorema 5.2.3. *Il numero medio di valutazioni di funzione affinché l'algoritmo Glob, avendo trovato un minimo locale $f(l_i)$, ne trovi uno nuovo (qualsiasi) è dato da:*

$$f_{i,*} = f_i \frac{1}{Prob_{i,*}}, \quad i = 1, ..., m-1, \tag{5.14}$$

con

$$Prob_{i,*} = \sum_{j=i+1}^{m} p_{i,j} + d_i \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \tag{5.15}$$

Dimostrazione. È sufficiente osservare che $Prob_{i,*}$ rappresenta la probabilità di trovare un nuovo minimo locale avendo trovato $f(l_i)$. Infatti, si trova un nuovo minimo se x_0 viene scelto in $A_{i,j}$, per $j = i+1, ..., m$, oppure viene scelto in $A_{0,j} - A_{i,j}$ con probabilità d_i. Tenendo conto che il numero di iterazioni necessarie a trovare un nuovo minimo è l'inverso della probabilità, dal suo prodotto con il numero di valutazioni per iterazione, dato dalla (5.9), si ottiene la (5.14). □

Lo studio della (5.14) è facile da realizzare. Determiniamo per quale valore di d_i si trova il minimo di tale funzione. Si ha per $f_{i,*},\ i = 1, ..., m-1$:

$$f_{i,*} = \frac{(k-1)\sum_{j=i+1}^{m} p_{i,j} + d_i(1 - \sum_{j=i+1}^{m} p_{i,j})(k-1) + 1}{\sum_{j=i+1}^{m} p_{i,j} + d_i(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j})}. \tag{5.16}$$

Derivando rispetto a d_i si ottiene:

$$f'_{i,*} = \frac{(k-1)(\sum_{j=i+1}^{m} p_{i,j})(1 - \sum_{j=i+1}^{m} p_{0,j}) - \sum_{j=i+1}^{m}(p_{0,j} - p_{i,j})}{(\sum_{j=i+1}^{m} p_{i,j} + d_i(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j}))^2}, \tag{5.17}$$

da cui si deduce che il segno della derivata non cambia per $d_i \in [0, 1]$. In particolare si trova, con semplici calcoli, che la derivata è maggiore o uguale a zero per

$$k \geq \frac{(\sum_{j=i+1}^{m} p_{0,j})(1 - \sum_{j=i+1}^{m} p_{i,j})}{(\sum_{j=i+1}^{m} p_{i,j})(1 - \sum_{j=i+1}^{m} p_{0,j})}. \tag{5.18}$$

La condizione (5.18) è importante perché correla le probabilità $p_{i,j}$ con il numero k di valutazioni di funzione eseguite ad ogni ricerca locale per scegliere il valore

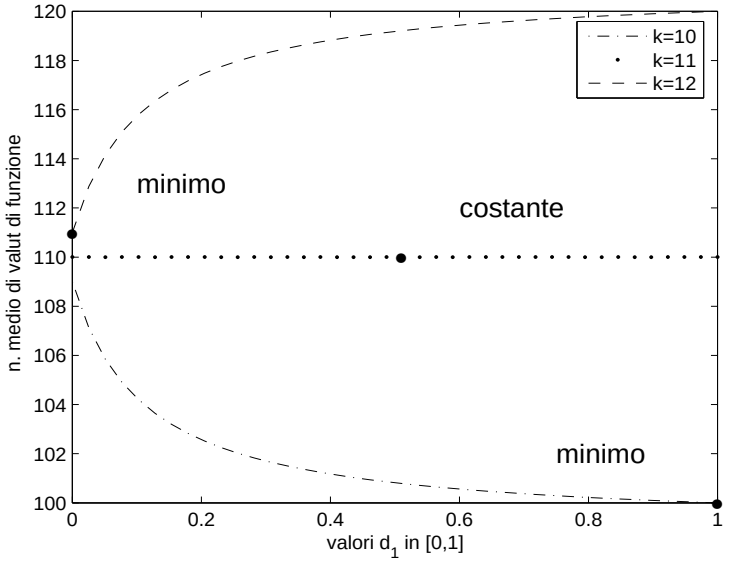

Figura 5.4: Numero medio di valutazioni di funzione necessarie per trovare un nuovo minimo locale a partire dall'ultimo minimo individuato

di d_i più vantaggioso: se la condizione è soddisfatta si dovrà prendere $d_i = 0$, altrimenti $d_i = 1$.

L'andamento di $f_{i,*}$ al variare di d_i è illustrato nella figura 5.4, in cui si è preso:

$$\sum_{j=i+1}^{m} p_{0,j} = 0.1, \quad \sum_{j=i+1}^{m} p_{i,j} = 0.01,$$

e in cui a k sono assegnati i valori 10, 11 e 12, rispettivamente.

Caso b) Numero finito di iterazioni

Si considerino valide le ipotesi 5.2.1. Si calcola innanzitutto la probabilità $Prob^{(n)}$ che effettuando n iterazioni nell'algoritmo *Glob*, l'ultimo minimo locale trovato sia globale. Si ricorda che T indica l'insieme di tutte le traiettorie possibili generate da *Glob* in cui l'ultimo minimo è globale.

Si ha:

Lemma 5.2.3. *La probabilità che, effettuando n iterazioni nell'algoritmo Glob, l'ultimo minimo locale sia globale è data da:*

$$Prob^{(n)}(n, d_1, ..., d_m) = \sum_{t \in T} Prob_t^{(n)}, \tag{5.19}$$

dove la $Prob_t^{(n)}$ *è calcolata nel lemma 5.2.1.*

Dimostrazione. Il lemma si ottiene semplicemente tenendo conto di tutte le traiettorie possibili. □

Si calcola ora $fun\_ev\_n(n, d_1, ..., d_m)$, numero medio di valutazioni di funzione nel caso in cui l'algoritmo *Glob* venga applicato per n iterazioni. Sia T' l'insieme di tutte le possibili traiettorie che l'algoritmo può individuare (in questo caso, l'ultimo minimo locale non è necessariamente globale) e sia $Prob_t$ la probabilità di generare la traiettoria $t \in T'$. Si dimostra il seguente teorema.

Teorema 5.2.4. *Il numero medio di valutazioni di funzione, applicando l'algoritmo Glob per n iterazioni, è dato da:*

$$fun\_ev\_n(n, d_1, ..., d_m) = \sum_{t \in T'} Prob_{0,i_1} \cdot Prob_{i_1,i_2} \cdot ... Prob_{i_{p-1},i_p} \cdot fun_t, \quad (5.20)$$

con fun_t *dato da:*

$$\begin{aligned} \sum_{\substack{j_1,...,j_p=0 \\ j_1+...+j_p=n-p}}^{n-p} & (kp + (kd_{i_1} + (1 - d_{i_1}))j_1 + (kd_{i_2} + (1 - d_{i_2}))j_2 + \\ + & ... + (kd_{i_p} + (1 - d_{i_p}))j_p) \cdot \\ \cdot & \ t_{i_1}^{j_1} \cdot t_{i_2}^{j_2} \cdot ... \cdot t_{i_p}^{j_p}. \end{aligned}$$

Dimostrazione. Si consideri una generica traiettoria $t = (i_1, ..., i_p)$ per la quale l'algoritmo, avendo trovato il punto di minimo locale l_{i_v}, $v = 1, ..., p$, non trova un nuovo minimo per j_v iterazioni. Si ha:

$$\sum_{v=1}^{p} j_v = n - p.$$

In tal caso la probabilità di generare la traiettoria t è data dalla:

$$\begin{aligned} & p_{0,i_1} \cdot (p_{i_1,i_2} + (p_{0,i_2} - p_{i_1,i_2})d_{i_1}) \cdot ... \cdot \\ & \cdot (p_{i_{p-i},i_p} + (p_{0,i_p} - p_{i_{p-1},i_p})d_{i_{p-1}}) \cdot \\ & \cdot t_{i_1}^{j_1} \cdot t_{i_2}^{j_2} \cdot ... \cdot t_{i_p}^{j_p}. \end{aligned} \quad (5.21)$$

Il numero medio di valutazioni di funzione è dato da:

$$kp + (kd_{i_1} + (1 - d_{i_1}))j_1 + (kd_{i_2} + (1 - d_{i_2}))j_2 + ... + (kd_{i_p} + (1 - d_{i_p}))j_p, \quad (5.22)$$

in cui kp è il numero di valutazioni di funzione necessarie per andare da un punto di minimo ad un altro, mentre $(kd_{i_v} + (1 - d_{i_v}))j_v$ rappresenta il numero medio di valutazioni di funzione utilizzate dall'algoritmo prima di trovare un nuovo minimo. Tenendo conto di tutte le possibili traiettorie si ha il lemma. □

A questo punto si può porre il seguente problema:

Problema 5.2.2. *Sia dato il problema 1.1.1 e siano noti i valori k, $p_{0,j}$ e $p_{i,j}$. Assegnato un numero reale positivo acc, con significato di accuratezza nei calcoli, trovare n^* e d_i^*, $i = 1, ..., m$, tali che:*

$$fun\_ev\_n(n^*, d_1^*, ..., d_m^*) = \min_{n,d_1,...,d_m} fun\_ev\_n(n, d_1, ..., d_m) \quad (5.23)$$

sotto la condizione:

$$Prob^{(n)}(n, d_1, ..., d_m) \geq 1 - acc. \quad (5.24)$$

Esempio 5.2.3.
Consideriamo il caso $m = 3$. In tal caso si ha possono avere 4 traiettorie e l'insieme T è definito da:

$$T = \{(3), (1,3), (2,3), (1,2,3)\}. \quad (5.25)$$

Dall'equazione (5.19), si trova per $Prob^{(n)}(n, d_1, d_2, d_3)$:

$$\begin{aligned}
&= p_{0,3} + \\
&+ p_{0,1}(p_{1,3} + (p_{0,3} - p_{1,3})d_1) \sum_{j=0}^{n-2} c_1^j + \\
&+ p_{0,2}(p_{2,3} + (p_{0,3} - p_{2,3})d_2) \sum_{j=0}^{n-2} c_2^j + \\
&+ p_{0,1}(p_{1,2} + (p_{0,2} - p_{1,2})d_1)(p_{2,3} + (p_{0,3} - p_{2,3})d_2) \cdot \\
&\cdot \sum_{\substack{j_1,j_2=0 \\ j_1+j_2 \leq n-3}}^{n-3} c_1^{j_1} c_2^{j_2},
\end{aligned}$$

con

$$c_1 = p_{0,1} + (p_{0,2} + p_{0,3} - p_{1,2} - p_{1,3})(1 - d_1), \quad (5.26)$$

$$c_2 = p_{0,1} + p_{0,2} + (p_{0,3} - p_{2,3})(1 - d_2). \quad (5.27)$$

L'insieme T' è dato da

$$T' = \{(1), (2), (3), (1,2), (1,3), (2,3), (1,2,3)\}. \quad (5.28)$$

Dall'equazione (5.20) si trova per $fun\_ev\_(n, d_1, d_2, d_3)$ la seguente espressione:

$$
\begin{aligned}
&= p_{0,1}c_1^{n-1}(k + e_1(n-1)) + \\
&+ p_{0,2}c_2^{n-1}(k + e_2(n-1)) + \\
&+ p_{0,3}(k + e_3(n-1)) + \\
&+ p_{0,1}(p_{1,2} + (p_{0,2} - p_{1,2})d_1) \sum_{\substack{j_1,j_2=0 \\ j_1+j_2=n-2}}^{n-2} c_1^{j_1}c_2^{j_2}(2k + e_1j_1 + e_2j_2) + \\
&+ p_{0,1}(p_{1,3} + (p_{0,3} - p_{1,3})d_1) \sum_{\substack{j_1,j_2=0 \\ j_1+j_2=n-2}}^{n-2} c_1^{j_1}(2k + e_1j_1 + e_3j_2) + \\
&+ p_{0,2}(p_{2,3} + (p_{0,3} - p_{2,3})d_2) \sum_{\substack{j_1,j_2=0 \\ j_1+j_2=n-2}}^{n-2} c_2^{j_1}(2k + e_2j_1 + e_3j_2) + \\
&+ p_{0,1}(p_{1,2} + (p_{0,2} - p_{1,2})d_1)(p_{2,3} + (p_{0,3} - p_{2,3})d_2) \cdot \\
&\cdot \sum_{\substack{j_1,j_2,j_3=0 \\ j_1+j_2+j_3=n-3}}^{n-3} c_1^{j_1}c_2^{j_2}(3k + e_1j_1 + e_2j_2 + e_3j_3),
\end{aligned}
$$

con c_1 e c_2 definiti in (5.26) e (5.27) rispettivamente e

$$
\begin{aligned}
e_1 &= kd_1 + (1 - d_1), \\
e_2 &= kd_2 + (1 - d_2), \\
e_3 &= kd_3 + (1 - d_3).
\end{aligned}
$$

Nella figura 5.5 si riporta il grafico di $fun\_ev\_n(n, d_1, d_2, d_3)$, con n scelto come il più piccolo intero tale che sia soddisfatto il vincolo (5.24) del problema 5.2.2 e assegnando a d_3 il valore zero. A k sono assegnati i valori 4 e 80 rispettivamente e $acc = 0.01$. Le probabilità usate sono assegnate nella tabella 5.2.
Analogamente all'esempio 5.2.1, i valori ottimali di d_1 e d_2 sono $d_1 = 0$, $d_2 = 0$ per $k = 4$ e $d_1 = 1$, $d_2 = 1$ per $k = 80$, ossia si trovano in corrispondenza dei vertici del cubo $[0, 1]^n$ di $\mathbb{R}^n$.

$p_{0,1} = 0.6$	$p_{0,2} = 0.2$	$p_{0,3} = 1 - p_{0,1} - p_{0,2}$
$p_{1,2} = 0.01$	$p_{1,3} = 0.04$	$p_{2,3} = 0.01$

Tabella 5.2: Valori delle probabilità per il calcolo di fun_ev_n

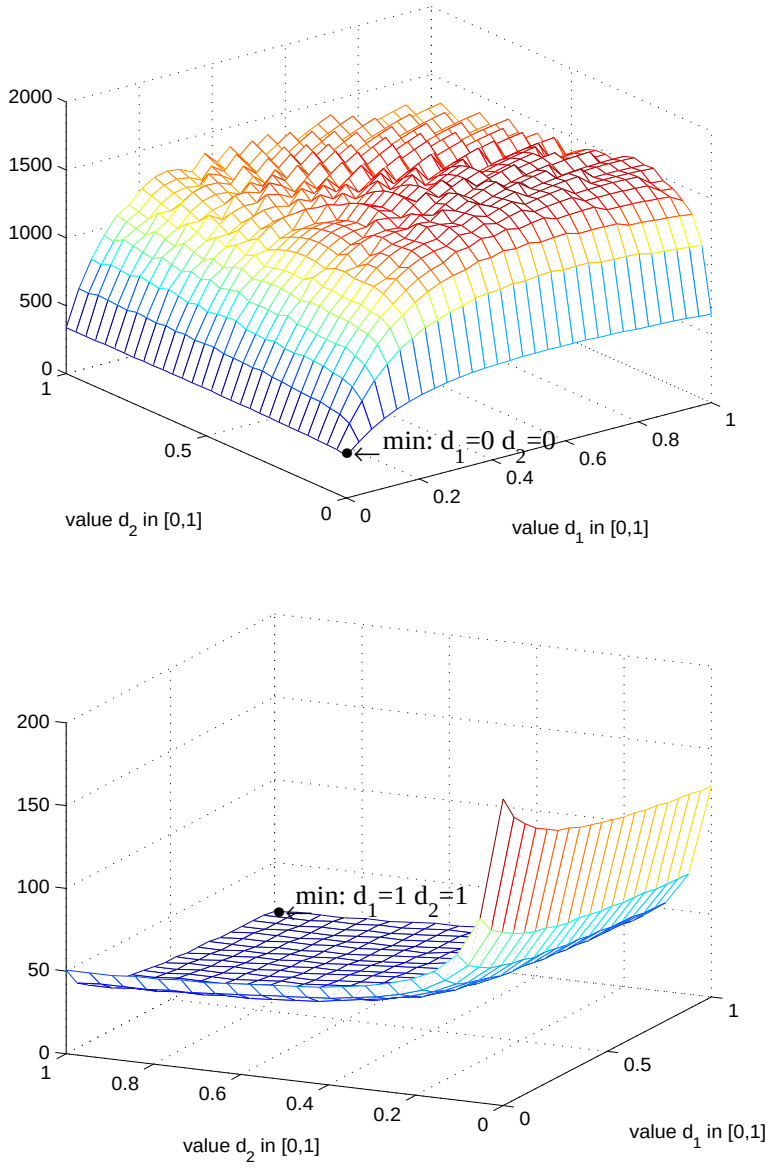

Figura 5.5: Numero medio di valutazioni di funzione con k=4 e k=80

Analisi Locale

Come nel caso a) si procede ora con un'analisi locale: si suppone di aver trovato un punto di minimo locale l_i e si calcola la probabilità $Prob_i^{(n)}$ di trovare un nuovo minimo locale (qualsiasi) disponendo di n iterazioni. Si ha:

Lemma 5.2.4.

$$Prob_i^{(n)} = 1 - t_i^n. \tag{5.29}$$

Dimostrazione. La probabilità $Prob_i^{(n)}$ è data da:

$$\left(\sum_{j=i+1}^{m} p_{i,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \cdot d_i \right) \cdot \sum_{j=0}^{n-1} t_i^j.$$

in cui il primo fattore si riferisce alla probabilità di trovare un nuovo minimo locale in una iterazione generica mentre la sommatoria si riferisce al numero di iterazioni nelle quali non viene trovato un nuovo minimo. L'applicazione della formula:

$$\sum_{s=0}^{n} t^s = \frac{1 - t^{n+1}}{1 - t} \tag{5.30}$$

ed il fatto che:

$$1 - t_i = \left(\sum_{j=i+1}^{m} p_{i,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \cdot d_i \right) \tag{5.31}$$

danno la (5.29).
La verifica che la probabilità $Prob_i^{(n)}$ è corretta risulta anche dalla seguente osservazione: la probabilità che, stando in l_i, non si trovi un nuovo minimo locale in n iterazioni è

$$\overline{Prob}_i^{(n)} = \left(\sum_{j=1}^{i} p_{0,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \cdot (1 - d_i) \right)^n, \tag{5.32}$$

e, chiaramente,

$$Prob_i^{(n)} + \overline{Prob}_i^{(n)} = 1.$$

□

Calcoliamo ora $fun\_ev\_n(n, d_i)$, numero medio di valutazioni di funzione quando si applica l'algoritmo per al più n iterazioni (a partire dal minimo in l_i) e si interrompe la sua esecuzione appena si trova un nuovo minimo locale. Si ha:

Lemma 5.2.5. *Il numero medio di valutazioni di funzione, per un'applicazione dell'algoritmo per al più n iterazioni ed interrompendo la sua esecuzione appena si trova un nuovo minimo locale, è dato da:*

$$fun\_ev\_n(n, d_i) = k \cdot (1 - t_i^n) + \frac{t_i(1 - t_i^n)(1 + d_i(k-1))}{1 - t_i}. \tag{5.33}$$

Dimostrazione. Nell'applicazione dell'algoritmo *Glob* per n iterazioni a partire dal punto di minimo locale l_i, possono verificarsi $n+1$ eventi $e = 1, ..., n+1$. I primi n eventi, $e = 1, ..., n$, indicano che l'algoritmo trova un nuovo minimo alla e-iesima iterazione (quindi per $e-1$ iterazioni si resta in l_i). L'ultimo evento, $e = n+1$, indica che in n iterazioni non viene trovato alcun minimo locale nuovo. La probabilità di ciascuno degli eventi $e = 1, ..., n$ è data da:

$$\left(\sum_{j=i+1}^{m} p_{i,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \cdot d_i \right) \cdot t_i^{e-1}, \tag{5.34}$$

e il numero medio di valutazioni di funzione per ciascun evento $e = 1, ..., n$, è:

$$\left(\sum_{j=i+1}^{m} p_{i,j} + \left(\sum_{j=i+1}^{m} p_{0,j} - \sum_{j=i+1}^{m} p_{i,j} \right) \cdot d_i \right) \cdot t_i^{e-1} \cdot \\ \cdot (k + (kd_i + (1 - d_i))(e-1)) \,. \tag{5.35}$$

I primi due fattori del prodotto in (5.35) si riferiscono alle probabilità di trovare un nuovo minimo locale in una iterazione generica o di non trovarne alcuno. Il terzo fattore dà il numero medio di valutazioni che vengono eseguite nell'evento e. La probabilità dell'evento $e = n+1$ è:

$$t_i^n,$$

quindi il numero medio di valutazioni di funzione per $e = n+1$ è il seguente:

$$t_i^n(kd_i + (1 - d_i))n.$$

Il numero medio totale di valutazioni di funzione sarà allora:

$$(1 - t_i) \sum_{e=1}^{n} t_i^{e-1} (k + (kd_i + (1 - d_i))(e-1)) + t_i^n(kd_i + (1 - d_i))n,$$

che può scriversi come:

$$(1 - t_i)k \sum_{e=0}^{n-1} t_i^e + (1 - t_i) \sum_{e=0}^{n-1} t_i^e (kd_i + (1 - d_i))e + t_i^n(kd_i + (1 - d_i))n.$$

Applicando le formule:

$$\sum_{s=0}^{n} t^s = \frac{1-t^{n+1}}{1-t}, \tag{5.36}$$

$$\sum_{s=0}^{n} st^s = \frac{-t^{n+1}(1+n(1-t))+t}{(1-t)^2}, \tag{5.37}$$

si trova:

$$k\cdot(1-t_i^n) + \frac{(kd_i+(1-d_i))(-t_i^n(1+(n-1)(1-t_i))+t_i)}{1-t_i} + \\ + t_i^n(kd_i+(1-d_i))n. \tag{5.38}$$

Da questa, con semplificazioni banali, si trova la (5.33). □

A questo punto operando allo stesso modo del problema 5.2.2, si può porre il problema:

Problema 5.2.3. *L'algoritmo Glob abbia individuato il punto di minimo locale l_i. Noti i valori k, $p_{0,j}$ e $p_{i,j}$, e assegnato un numero reale positivo acc, con significato di accuratezza nei calcoli, trovare n^* e d_i^* tali che:*

$$fun\_ev\_n(n^*, d_i^*) = \min_{n,d_i} k\cdot(1-t_i^n) + \frac{t_i(1-t_i^n)(1+d_i(k-1))}{1-t_i},$$

$$\textit{sotto la condizione} \qquad 1-t_i^n \geq 1-acc. \tag{5.39}$$

Nella figura 5.6 si riporta il grafico di $fun\_ev\_n(n, d_i)$ con $acc = 0.001$,

$$\sum_{j=1}^{i} p_{0,j} = 0.9, \quad \sum_{j=i+1}^{m} p_{0,j} = 0.1, \quad \sum_{j=i+1}^{m} p_{i,j} = 0.01,$$

ed n scelto in modo da soddisfare il vincolo (5.39). I grafici mettono in evidenza ancora una volta che il minimo viene raggiunto per $d_i = 0$ oppure per $d_i = 1$.

Si studia ora il problema 5.2.3 in modo da individuarne la soluzione. Assumendo che il vincolo (5.39) sia una uguaglianza (la soluzione ottimale n^* è data dal più piccolo intero per cui il vincolo è soddisfatto), si ha:

$$1-t_i^n = 1-acc \tag{5.40}$$

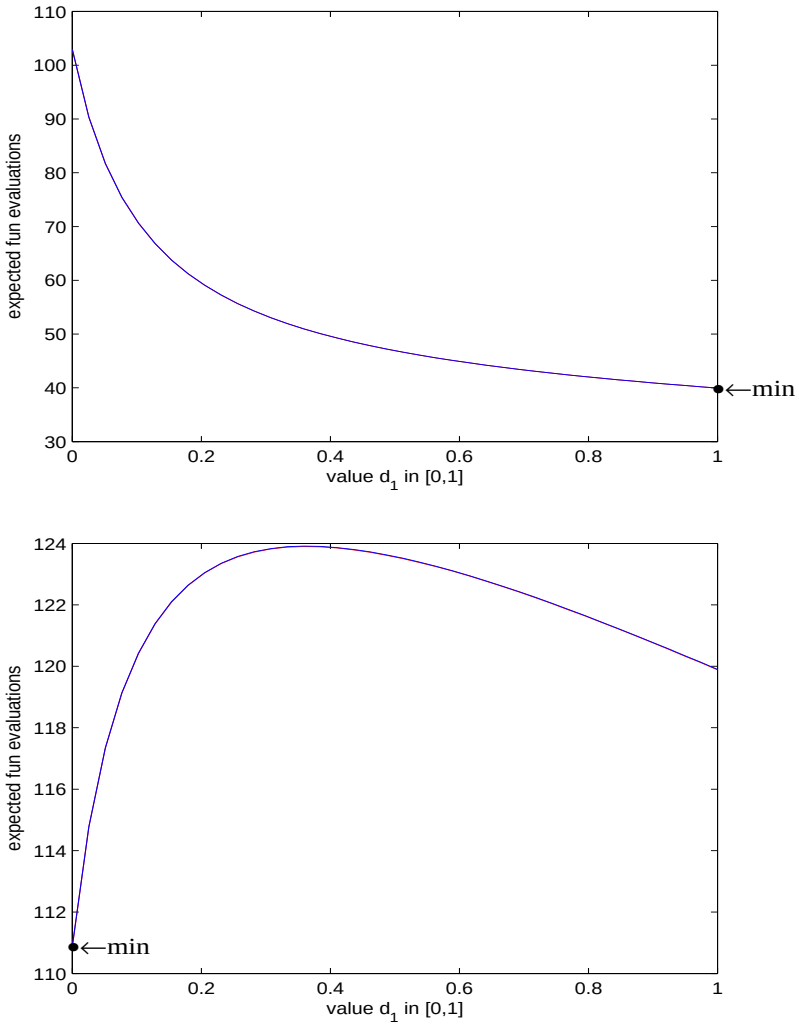

Figura 5.6: Numero medio di valutazioni di funzione con k=4 e k=12

e la funzione da minimizzare diventa:

$$g(d_i) = k \cdot (1 - acc) + \frac{t_i(1 - acc)(1 + d_i(k-1))}{1 - t_i}, \tag{5.41}$$

in cui t_i dipende da d_i. La derivata della (5.41), dopo banali semplificazioni, diventa un'equazione algebrica di secondo grado in d_i :

$$\frac{-(p_2 - p_3)^2(k-1)d_i^2 - 2p_3(k-1) \cdot (p_2 - p_3)d_i - p_2 + kp_3 - p_3^2(k-1)}{(1 - t_i)^2}, \tag{5.42}$$

in cui p_2 e p_3 sono definiti da:

$$p_2 = \sum_{j=i+1}^{m} p_{0,j}, \quad p_3 = \sum_{j=i+1}^{m} p_{i,j}. \tag{5.43}$$

L'analisi dei segni dei coefficienti della (5.42) permette di affermare che non esiste un minimo locale all'interno dell'intervallo $[0, 1]$ e pertanto il minimo di $g(d_i)$ si ottiene agli estremi, vale a dire per $d_i = 0$ o per $d_i = 1$. Si ha:

$$g(0) = k \cdot (1 - acc) + \frac{(1 - p_3)(1 - acc)}{p_3}, \tag{5.44}$$

$$g(1) = k \cdot (1 - acc) + \frac{(1 - p_2)(1 - acc)k}{p_2}. \tag{5.45}$$

La disequazione $g(0) \leq g(1)$ (cioè l'ipotesi che g sia crescente) dà:

$$\frac{1 - p_3}{p_3} \leq \frac{k(1 - p_2)}{p_2}; \tag{5.46}$$

pertanto si ritrova la (5.18), ossia la stessa relazione ottenuta nel caso a).

5.3 Un nuovo algoritmo per l'ottimizzazione globale

Nel risolvere i problemi reali, in generale non si conoscono né i valori $p_{0,j}$ e $p_{i,j}$ né il loro numero; pertanto la scelta di $d_1, d_2, ..., d_m$ nell'ottimizzazione della funzione fun_ev in (5.7) oppure di fun_ev_n nel problema 5.2.2, non può essere presa in considerazione. Al contrario si può utilizzare l'analisi locale vista sia per il caso a) (infinite iterazioni), sia per il caso b) (numero finito di iterazioni). Mediante delle stime dei valori:

$$p_2 = \sum_{j=i+1}^{m} p_{0,j}, \; p_3 = \sum_{j=i+1}^{m} p_{i,j}, \tag{5.47}$$

che compaiono sia nell'espressione (5.14) della $f_{i,*}$, sia in fun_ev_n nel problema 5.2.3, possiamo mettere a punto un criterio per la scelta dei valori d_i, $i = 1, \ldots, m$, nell'algoritmo *Glob*. In particolare dalla (5.18) e dalla (5.46) risulta:

$$d_i = \begin{cases} 0 & \text{se } k > (p_2 \cdot (1 - p_3))/(p_3 \cdot (1 - p_2)), \\ 1 & \text{altrimenti,} \end{cases} \tag{5.48}$$

in cui p_2, p_3 e k sono approssimati come segue:

$$\begin{aligned} p_2 &= 1/(\text{numero di ricerche effettuate}), \\ p_3 &= 1/(\text{numero di iterazioni effettuate finora}), \\ k &= \text{numero medio di valutazioni di funzione nelle ricerche locali.} \end{aligned}$$

Pertanto l'istruzione:

$$\textbf{if } f(x_0) \leq fl_i \quad \textbf{oppure} \quad (f(x_0) > fl_i \textbf{ e } rand(1) < d_i)$$

dell'algoritmo *Glob* è sostituita dall'istruzione:

$$\textbf{if } f(x_0) \leq fl_i \quad \textbf{oppure yes\_box}() = 1 \tag{5.49}$$

in cui **yes_box**() è un algoritmo che restituisce 0 oppure 1, definito come segue:

Algoritmo 5.3.1 (yes_box()).

$p_2 = 1/$ (numero di ricerche effettuate);
$p_3 = 1/$ (numero di iterazioni effettuate finora);
$k =$ numero medio di valutazioni di funzione nelle ricerche locali;
if $p_2 = 1$
  $p_2 = 2 \cdot p_3$;
end if
if $p_2 < 1$
  ratio=$(p_2 \cdot (1 - p_3))/(p_3 \cdot (1 - p_2))$;
else
  ratio=inf;
end if
if k>ratio
  yes=0;
else
  yes=1;
end if
end

In seguito, l'algoritmo generale *Glob* in cui i parametri vengono definiti come descritto sopra, verrà indicato con *Glob_mod*.

5.4 Risultati Numerici

Si presentano ora i risultati numerici ottenuti risolvendo vari problemi test proposti in letteratura. Si sono presi in considerazione 2 gruppi di problemi: fun_test_1 e

fun_test_2. Il primo gruppo contiene due problemi definiti in (5.50) ed è derivato dalle classi di funzioni test definite in [26] e realizzabili anche mediante il software GKLS (cfr. [23]); il secondo gruppo è stato utilizzato da Dekkers e Aarts in [12] e viene descritto nell'Appendice.

$$f(x) = \begin{cases} \left(\frac{2}{\rho^2}\frac{\langle x-M,-M\rangle}{\|x-M\|} - \frac{2}{\rho^3}(\|M\|^2+2)\right)\|x-M\|^3 + \\ + \left(1 - \frac{4}{\rho}\frac{\langle x-M,-M\rangle}{\|x-M\|} + \frac{3}{\rho^2}(\|M\|^2+2)\right)\|x-M\|^2, & \text{if } x \in B, \\ \\ \|x\|^2 + 2, & \text{if } x \notin B, \end{cases} \tag{5.50}$$

con $S = \{x \in \mathbb{R}^4 \mid 0 \le x_i \le 1,\ i = 1,\dots,4\}$, $B = \{x \in \mathbb{R}^4 \mid \|x - M\| \le \rho\}$ e con $M = (1,1,1,1)$, $\rho = 0.67$ per il problema test $artif\_1$ e $M = (1.2, 1.2, 1.2, 1.2)$, $\rho = 1.36$ per $artif\_2$. (Entrambe le funzioni test hanno un minimo locale in $T = (0,0,0,0)$, $f(T) = 2$. La differenza principale tra le due funzioni consiste nel fatto che il gradiente relativo al minimo globale di $artif\_1$ è nullo mentre il gradiente nel minimo globale di $artif\_2$ è diverso da zero, trovandosi tale minimo sulla frontiera di S.)

La ricerca locale è stata realizzata mediante la funzione *funmincon* dell'Optimization Toolbox di MATLAB. Questa funzione usa un metodo *quasi-Newton* con la formula *BFGS* per aggiornare l'approssimazione dell'hessiano della funzione da minimizzare. Inoltre usa la formula *quadcubic* per la ricerca lungo una direzione.

Nelle tabelle 5.3 e 5.4 si riportano i risultati ottenuti nel risolvere i problemi test in fun_test_1 e fun_test_2 mediante l'algoritmo *Glob*: in questo caso, l'algoritmo 5.2.1 è stato eseguito con sei valori del *parametro probabilità* d_i distribuiti uniformemente nell'intervallo $[0, 1]$.

Nelle tabelle 5.5 e 5.6 si riportano i risultati per la risoluzione degli stessi problemi applicando l'algoritmo *Glob_mod* proposto nella presente tesi, con d_i definiti in (5.48). Da notare che i valori riferiti al numero di valutazioni di funzione (fun_eval) e al numero di ricerche locali (loc_search) sono stati ottenuti come valori medi di 100 esecuzioni dell'algoritmo che ha iterato per 1000 iterazioni. Infatti, poiché l'algoritmo utilizza tecniche casuali, i risultati riferiti a poche esecuzioni non sarebbero stati significativi.

Nei casi in cui il minimo globale (noto) è stato raggiunto, l'errore commesso nella sua individuazione varia tra 10^{-5} e 10^{-15} (si vedano le tabelle 5.7-5.10).

Dall'analisi dei risultati è evidente che, per le classi di problemi prese in esame, la scelta del parametro $d_i = 0$ nell'algoritmo *Glob* implica un minore numero di valutazioni di funzione nell'esecuzione dell'algoritmo, tuttavia in 5 casi su 13 il minimo globale non viene individuato (cioè in almeno un caso su 100 l'algoritmo non converge al minimo globale). Il valore $d_i = 0.2$ risolve tutti i problemi test. Ciò permette di affermare che i valori ottimali di d_i si collocano nell'intervallo $[0, 0.2]$. Il nuovo algoritmo *Glob_mod* risolve 12 problemi su 13 ed il suo costo

computazionale è, ad eccezione di un caso, sempre inferiore al caso $d_i = 0.2$. Spesso la differenza in termini di valutazione di funzione è notevole a favore del nuovo algoritmo.
Pertanto si può concludere che la tecnica introdotta nella tesi per approssimare il valore di d_i ottimale, è promettente e può portare alla messa a punto di algoritmi efficienti.

			d_i					
probl. no.	test problem		0.0	0.2	0.4	0.6	0.8	1
1	$artif\_1$	fun_eval	NO	1682	2364	3035	3731	4417
		loc_search	NO	200	400	601	802	1001
2	$artif\_2$	fun_eval	NO	1957	2901	3846	4792	5761
		loc_search	NO	201	401	600	802	1001

Tabella 5.3: Risultati dell'algoritmo *Glob* per fun_set_1, con valori d_i distribuiti uniformemente in $[0, 1]$

probl.	test		d_i					
no.	problem		0.0	0.2	0.4	0.6	0.8	1
1	14_GP	fun_eval	1050	8667	16155	23871	31376	38999
		loc_search	1	203	399	602	801	1001
2	08_BRAN	fun_eval	1025	5821	10638	15514	20258	25031
		loc_search	1	201	401	604	801	1001
3	18_HAR3	fun_eval	1037	6955	12699	18613	24484	30359
		loc_search	1	203	399	601	801	1001
4	19_HAR6	fun_eval	NO	11897	22686	33428	44181	55069
		loc_search	NO	202	402	600	799	1001
5	63_SHE5	fun_eval	1007	2221	3453	4673	5897	7115
		loc_search	1	200	402	602	802	1001
6	64_SHE7	fun_eval	1007	2222	3446	4666	5891	7111
		loc_search	1	200	401	602	802	1001
7	65_SH10	fun_eval	1007	2225	3448	4668	5896	7104
		loc_search	1	201	401	602	802	1001
8	66_SHU	fun_eval	1082	6356	11744	17002	22290	27576
		loc_search	3	202	404	603	803	1001
9	21_LEV1	fun_eval	NO	9063	17003	25085	33040	40931
		loc_search	NO	202	400	602	802	1001
10	31_LEV3	fun_eval	NO	21626	41949	62061	82722	102994
		loc_search	NO	202	402	601	802	1001
11	02_AL22	fun_eval	1036	7752	14516	21192	27872	34688
		loc_search	1	201	402	600	800	1001

Tabella 5.4: Risultati dell'algoritmo *Glob* per fun_set_2, con valori d_i distribuiti uniformemente in $[0, 1]$

probl. no.	test problem		
1	$artif\_1$	fun_eval	1779
		loc_search	229
2	$artif\_2$	fun_eval	1835
		loc_search	175

Tabella 5.5: Risultati dell'algoritmo $Glob\_mod$ per fun_set_1, con valori d_i definiti nella (5.48)

probl. no.	test problem		
1	14_GP	fun_eval	2031
		loc_search	27
2	08_BRAN	fun_eval	1995
		loc_search	41
3	18_HAR3	fun_eval	2011
		loc_search	34
4	19_HAR6	fun_eval	2063
		loc_search	20
5	63_SHE5	fun_eval	1869
		loc_search	142
6	64_SHE7	fun_eval	1869
		loc_search	142
7	65_SH10	fun_eval	1869
		loc_search	143
8	66_SHU	fun_eval	2005
		loc_search	38
9	21_LEV1	fun_eval	2039
		loc_search	26
10	31_LEV3	fun_eval	NO
		loc_search	NO
11	02_AL22	fun_eval	2022
		loc_search	30

Tabella 5.6: Risultati dell'algoritmo $Glob\_mod$ per fun_set_2, con valori d_i definiti nella (5.48)

		d_i					
probl. no.	test problem	0.0	0.2	0.4	0.6	0.8	1
1	$artif\_1$	NO	7.5e-14	5.1e-14	1.9e-14	4.5e-14	1.6e-14
2	$artif\_2$	NO	5.7e-15	5.7e-15	5.7e-15	5.7e-15	5.7e-15

Tabella 5.7: Errore per fun_set_1 e d_i scelti uniformemente in $[0, 1]$

		d_i					
probl. no.	test problem	0.0	0.2	0.4	0.6	0.8	1
1	14_GP	3.5e-14	4.1e-14	3.2e-14	3.9e-14	3.2e-14	5.5e-14
2	08_BRAN	3.6e-07	3.6e-07	3.6e-07	3.6e-07	3.6e-07	3.6e-07
3	18_HAR3	2.1e-06	2.1e-06	2.1e-06	2.1e-06	2.1e-06	2.1e-06
4	19_HAR6	NO	2.0e-06	2.0e-06	2.0e-06	2.0e-06	2.0e-06
5	63_SHE5	4.4e-06	4.4e-06	4.4e-06	4.4e-06	4.4e-06	4.4e-06
6	64_SHE7	3.3e-05	3.3e-05	3.3e-05	3.3e-05	3.3e-05	3.3e-05
7	65_SH10	2.9e-05	2.9e-05	2.9e-05	2.9e-05	2.9e-05	2.9e-05
8	66_SHU	1.2e-06	1.2e-06	1.2e-06	1.2e-06	1.2e-06	1.2e-06
9	21_LEV1	NO	9.0e-14	1.5e-13	1.1e-13	1.4e-13	1.6e-13
10	31_LEV3	NO	8.6e-13	1.0e-12	1.1e-12	1.1e-12	1.0e-12
11	02_AL22	2.3e-09	2.3e-09	2.3e-09	2.3e-09	2.3e-09	2.3e-09

Tabella 5.8: Errore per fun_set_2 e d_i scelti uniformemente in $[0, 1]$

probl. no.	test problem	error
1	$artif\_1$	2.0e-14
2	$artif\_2$	5.7e-15

Tabella 5.9: Errore per fun_set_1 e d_i definiti nella (5.48)

probl. no.	test problem	error
1	14_GP	4.9e-14
2	08_BRAN	3.6e-07
3	18_HAR3	2.1e-06
4	19_HAR6	2.0e-06
5	63_SHE5	4.4e-06
6	64_SHE7	3.3e-05
7	65_SH10	2.9e-05
8	66_SHU	1.2e-06
9	21_LEV1	1.6e-13
10	31_LEV3	NO
11	02_AL22	2.3e-09

Tabella 5.10: Errore per fun_set_2 e d_i definiti nella (5.48)

5.5 Conclusioni

Nella presente tesi si è presentato un nuovo metodo, basato su ricerche locali, per la determinazione di un punto di minimo globale di una funzione reale. In particolare si sono poste le basi teoriche per la costruzione di nuovi algoritmi che riducono il numero totale di valutazioni di funzione necessarie a trovare il minimo globale della funzione obiettivo. Sulla base dei risultati teorici si è data una prima implementazione del metodo proposto. L'algoritmo risultante è stato testato su numerose funzioni test utilizzate nella letteratura. I risultati ottenuti, confrontati con versioni dell'algoritmo in cui non si è effettuata alcuna analisi per la riduzione del numero di valutazioni di funzione, hanno dimostrato che il nuovo approccio è positivo.

Il lavoro portato avanti ha messo in evidenza vari aspetti da approfondire; in particolare:

- individuazione, tramite le relazioni esposte nella tesi, di proprietà generali che "a priori" consentono di stabilire il comportamento ottimale di un algoritmo per una determinata famiglia di problemi;
- ricerca dei parametri ottimali da assegnare ad un algoritmo per l'implementazione del metodo proposto;
- confronto tra il nuovo algoritmo e i vari algoritmi presenti nella letteratura mediante un criterio comune di stop.

Appendice

Si riportano successivamente le funzioni test proposte da Dixon e Szegö in [17] e da Aluffi-Pentini et al. in [3], e utilizzate da Dekkers e Aarts in [12].

Funzioni test di Dixon e Szegö

GP (Goldstein e Price)

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot \\ \cdot[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$$
$$S = \{x \in \mathbb{R}^2 \mid -2 \leq x_i \leq 2,\ i = 1, 2\}, \quad x_{\min} = (0, -1),\ f(x_{\min}) = 3.$$

La funzione ha 3 minimi locali, oltre al minimo globale.

BRAN (Branin)

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e,$$

dove $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $d = 6$, $e = 10$, $f = 1/(8\pi)$.

$$S = \{x \in \mathbb{R}^2 \mid -5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15\},$$
$$x_{\min} = (-\pi, 12.275);\ (\pi, 2.275);\ (3\pi, 2.475),\ f(x_{\min}) = 5/(4\pi).$$

Non esistono altri minimi.

HAR3 e HAR6 (famiglia di funzioni di Hartmann)

$$f(x) = -\sum_{i=1}^{m} c_i \exp\left(-\sum_{j=1}^{n} a_{ij}(x_i - p_{ij})^2\right),$$
$$S = \{x \in \mathbb{R}^n \mid 0 \leq x_i \leq 1,\ 1 \leq i \leq n\}.$$

$x_{\min} = (0.114, 0.556, 0.852),\ f(x_{\min}) = -3.86,$ se $n = 3$;
$x_{\min} = (0.201, 0.150, 0.477, 0.275, 0.311, 0.657),\ f(x_{\min}) = -3.32,$ se $n = 6$.
Queste funzioni hanno 4 minimi locali, $x_{\text{loc}} \approx (p_{i1}, \ldots, p_{in}), f(x_{\text{loc}}) \approx -c_i$.

i	a_{ij}			c_i	p_{ij}		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

Tabella 5.11: HAR3 ($n = 3$ e $m = 4$)

i	a_{ij}						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

p_{ij}					
0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Tabella 5.12: HAR6 ($n = 6$ e $m = 4$)

SHE5, SHE7 e SH10 (famiglia di funzioni di Shekel)

$$f(x) = -\sum_{i=1}^{m}((x - a_i)^T(x - a_i) + c_i)^{-1},$$

dove la dimensione è $n = 4$ e $m = 5, 7, 10$ per SHE5, SHE7 e SH10 rispettivamente; $x = (x_1, \ldots, x_n)^T$ e $a_i = (a_{i1}, \ldots, a_{in})^T$.

i	a_{ij}				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Tabella 5.13: SHE5, SHE7 e SH10

$S = \{x \in \mathbb{R}^4 \mid 0 \le x_j \le 10,\ 1 \le j \le 4\}$. Le funzioni SHE5, SHE7 e SH10 hanno rispettivamente 5, 7 e 10 minimi locali, aventi i seguenti valori:

$$x_{\text{loc}} \approx a_i, \quad f(x_{\text{loc}}) \approx 1/c_i, \quad i = 1, \ldots, m.$$

Funzioni test di Aluffi-Pentini et al.

SHU (funzione di Shubert bidimensionale)

$$f(x_1, x_2) = \left\{\sum_{i=1}^{5} i \cos[(i+1)x_1 + 1]\right\}\left\{\sum_{i=1}^{5} i \cos[(i+1)x_2 + 1]\right\},$$
$$S = \{x \in \mathbb{R}^2 \mid -10 \le x_i \le 10,\ i = 1, 2\}.$$

Questa funzione ha 760 minimi locali, di cui 18 sono globali.

LEV1 (funzione di Levi no.1)

$$f(x) = \frac{\pi}{n}\left\{k_1 \sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - k_2)^2[1 + k_1 \sin^2(\pi y_{i+1})] + (y_n - k_2)^2\right\},$$

dove $y_i = 1 + \frac{1}{4}(x_i + 1)$, $k_1 = 10$ e $k_2 = 1$.

$S = \{x \in \mathbb{R}^3 \mid -10 \le x_i \le 10,\ i = 1, 2, 3\}$, $x_{\min} = (1, 1, 1)$, $f(x_{\min}) = 0$.

Questa funzione ha circa 5^3 minimi locali.

LEV3 (funzione di Levi no.3)

$$f(x) = k_3\Bigg\{\sin^2(\pi k_4 x_1) + \sum_{i=1}^{n-1}(x_i - k_5)^2[1 + k_6 \sin^2(\pi k_4 x_{i+1})] + \\ +(x_n - k_5)^2[1 + k_6 \sin^2(\pi k_7 x_n)]\Bigg\},$$

dove $k_3 = 0.1$, $k_4 = 3$, $k_5 = 1$, $k_6 = 1$ e $k_7 = 2$.

$S = \{x \in \mathbb{R}^5 \mid -5 \le x_i \le 5,\ i = 1, \ldots, 5\}$, $x_{\min} = (1, 1, 1, 1, 1)$, $f(x_{\min}) = 0$.

Questa funzione ha circa 15^5 minimi locali.

Al22 (funzione di Aluffi-Pentini)

$$f(x) = 10^k x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^l (x_1^2 + x_2^2)^4,$$

dove $k = 5$ e $l = -5$.

$$S = \{x \in \mathbb{R}^2 \mid -20 \leq x_i \leq 20,\ i = 1, 2\},$$
$$x_{\min} = (0, 15);\ (0, -15),\ f(x_{\min}) = -24775.$$

Questa funzione ha un minimo locale nell'origine.

Ringraziamenti

Al termine di questi tre anni di Dottorato, desidero esprimere la mia gratitudine a tutte le persone che, direttamente o indirettamente, hanno contribuito alla realizzazione di questo lavoro.

Ringrazio innanzitutto il mio Tutore, il Prof. Marco Gaviano, per tutti gli insegnamenti, per l'aiuto nell'impostazione e nella stesura della tesi e per l'attenzione con cui ha sempre seguito il mio lavoro di ricerca, nonostante i numerosi impegni.

Un ringraziamento particolare va al Coordinatore del Dottorato in Matematica, il Prof. Giovanni Porru, che mi ha permesso di portare a termine il ciclo di Dottorato grazie ad un contributo ottenuto dalla Fondazione Banco di Sardegna.

Un ringraziamento affettuoso va alla Dott.ssa Daniela Lera, per i validi consigli e l'appoggio costante fin dal periodo della tesi di laurea.

Grazie alla Prof.ssa Alba Dolci, per tutti i chiarimenti su problemi di Calcolo delle Probabilità, per l'aiuto nei primi tempi del Dottorato e per l'interesse con cui ha sempre seguito il mio lavoro.

Grazie anche al Dott. Claudio Estatico, per alcuni utili suggerimenti sull'impostazione finale della tesi.

Grazie ai miei cari e a tutti i miei amici, per avermi sempre incoraggiato nei momenti difficili, e ai miei colleghi, in particolare i ragazzi dello studio e coloro che hanno condiviso con me le lunghe giornate in Dipartimento, per aver reso questi anni davvero speciali.

Grazie a tutti.
Maristella

Bibliografia

[1] E.H.L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. J. Wiley & Sons, 1989.

[2] J. Abaffy, A. Dolci, and M. Gaviano. Bayesian stopping rules for improvement of local minima algorithms in global optimization. *Optimization*, 30: 215–226, 1994.

[3] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications*, 47: 1–16, 1985.

[4] F. Archetti and B. Betrò. A priori analysis of deterministic strategies for global optimization. In L.C.W. Dixon and G.P. Szegö, editors, *Towards Global Optimization 2*, pages 31–48. North-Holland, Amsterdam, 1978.

[5] J. Barhen, V. Protopopescu, and D. Reister. TRUST: a deterministic algorithm for global optimization. *Science*, 276: 1094–1097, 1997.

[6] R.W. Becker and G.V. Lago. A global optimization algorithm. In *Proceedings of the 8-th Allerton Conference on Circuits and Systems Theory*, 1970.

[7] C. Bélisle. Slow hit-and-run sampling. *Statist. Probab. Lett.*, 47(1): 33–43, 2000.

[8] C.J.P. Bélisle. Convergence theorems for a class of simulated annealing algorithms on R^d. *Journal of Applied Probability*, 29: 885–892, 1992.

[9] A. Bernasconi and B. Codenotti. *Introduzione alla complessità computazionale*. Springer, 1998.

[10] C.G.E. Boender, R.J. Caron, and A.H.G. Rinnooy Kan. Shake-and-bake algorithms for generating uniform points on the boundary of bounded polyhedra. *Operations Research*, 39(6): 945–954, 1991.

[11] L. Daboni. *Calcolo delle probabilità ed elementi di statistica*. UTET, 1974.

[12] A. Dekkers and E. Aarts. Global optimization and simulated annealing. *Mathematical Programming*, 50: 367–393, 1991.

[13] J.E. Dennis and R.B. Schnabel. A view of unconstrained optimization. In G.L. Nemhauser and oth., editors, *Handbooks in OR & MS*, volume 1, chapter I, pages 1–72. Elsevier Science Publishers B.V. (North-Holland), 1989.

[14] L.C.W. Dixon. Global optima without convexity. Technical report, Numerical Optimization Centre, Hatfield Polytechnic, Hatfield, England, 1978.

[15] L.C.W. Dixon and G.P. Szegö. *Towards Global Optimization 2*. North-Holland, Amsterdam, 1978.

[16] L.C.W. Dixon and G.P. Szegö. *Towards Global Optimization*. North-Holland, Amsterdam, 1975.

[17] L.C.W. Dixon and G.P. Szegö. The global optimization problem: an introduction. In L.C.W. Dixon and G.P. Szegö, editors, *Towards Global Optimization 2*, pages 1–15. North-Holland, Amsterdam, 1978.

[18] C.C.Y. Dorea. On the efficiency of a continuous version of the simulated annealing algorithm. *Statistics and Probability Letters*, 31: 247–253, 1997.

[19] M. Dyer and A. Frieze. Random walks, totally unimodular matrices, and a randomised dual simplex algorithm. *Mathematical Programming*, 64(1): 1–16, 1994.

[20] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *Journal of the Association for Computing Machinery*, 38(1): 1–17, 1991.

[21] F. Fontanella and A. Pasquali. *Calcolo numerico: metodi e algoritmi*, volume 2. Pitagora Editrice Bologna, 1982.

[22] M. Gaviano. Necessary and sufficient conditions for the convergence of an algorithm in unconstrained minimization. In L.C.W. Dixon and G.P. Szegö, editors, *Towards Global Optimization*. North-Holland, Amsterdam, 1975.

[23] M. Gaviano, D. E. Kvasov, D. Lera, and Y. D. Sergeyev. Software for generation of classes of test functions with known local and global minima for global optimization. *ACM, Transaction on Mathematical Software*, 29(4): 469–480, 2003.

[24] M. Gaviano and D. Lera. A global minimization algorithm for Lipschitz functions. *Optimization Letters, to appear.*

[25] M. Gaviano and D. Lera. Solution of global optimization problems by improvement of local minima algorithms. *Rend. Sem. Fac. Sci. Università di Cagliari*, 65(1): 25–62, 1995.

[26] M. Gaviano and D. Lera. Test functions with variable attraction regions for global optimization problems. *Journal of Global Optimization*, 13(2): 207–223, 1998.

[27] M. Gaviano and D. Lera. A complexity analysis of local search algorithms in global optimization. *Optimization Methods Software*, 17(1): 113–127, 2002.

[28] M. Gaviano and D. Lera. Complexity of general continuous minimization problems: a survey. *Optimization Methods and Software*, 20(4/5): 525–544, 2005.

[29] R.P. Ge. The theory of the filled function method for finding a global minimizer of a nonlinearly constrained minimization problem. *SIAM Conf. on Num. Opt., Boulder, CO, J.of Computational Math.*, 5(3): 1–9, 1984.

[30] C. Guus, E. Boender, and H.E. Romeijn. Stochastic methods. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers, The Netherlands, 1995.

[31] E. Hansen. *Global optimization using interval analysis.* Marcel Dekker, Inc., 1992.

[32] P. Hansen and B. Jaumard. Lipschitz optimization. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 407–493. Kluwer Academic Publishers, The Netherlands, 1995.

[33] J.B. Hiriart-Urruty. Conditions for global optimality. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 1–26. Kluwer Academic Publishers, The Netherlands, 1995.

[34] R. Horst, P.M. Pardalos, and N.V. Thoai. Introduction to global optimization. In *Nonconvex Optimization and its Applications*, volume 3. Kluwer Academic Publishers, Dordrecht, 1995.

[35] L. Ingber. Simulated Annealing: practice versus theory. *Mathematical and Computer Modelling*, 18(11): 29–57, 1993.

[36] L. Ingber. Adaptive Simulated Annealing ASA: lesson learned. *Control and Cybernetics*, 25(1): 33–54, 1996.

[37] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220: 671–680, 1983.

[38] H.J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Transactions of ASME, Series D, Journal of Basic Engineering*, 86: 97–105, 1964.

[39] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Practice.* Reidel, Dordrecht, 1987.

[40] A.V. Levi and S. Gomez. The tunneling algorithm applied to global optimization. In P.T. Boggs, R.H. Byrd, and R.B. Schnabel, editors, *Numerical Optimization*, pages 213–244. SIAM, 1984.

[41] A.V. Levi and A. Montalvo. The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing*, 6(15): 15–29, 1985.

[42] M. Locatelli. Convergence properties of simulated annealing for continuous global optimization. *Journal of Applied Probability*, 33: 1127–1140, 1996.

[43] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, 104: 121–133, 2000.

[44] M. Locatelli. Simulated annealing algorithms for continuous global optimization. In *Handbook of Global Optimization*, pages 179–229. Kluwer Academic Publishers, Dordrecht, 2002.

[45] L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4): 985–1005, 2006.

[46] S. Lucidi and M. Piccioni. Random tunneling by means of acceptance-rejection sampling for global optimization. *Journal of Optimization Theory and Applications*, 62(2): 255–277, 1989.

[47] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, and A.H. Teller. Equation of state calculations by fast computer machines. *The Journal of Chemical Physics*, 21: 1087–1092, 1953.

[48] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, N.Y., 1966.

[49] A.S. Nemirovsky and D.B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley and Sons, Chichester, 1983.

[50] E.M. Oblow. SPT: a stochastic tunneling algorithm for global optimization. *Journal of Global Optimization*, 20: 195–212, 2001.

[51] S.A. Pijavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12: 57–67, 1972.

[52] J.D. Pintér. Global optimization in action. In *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 1996.

[53] H. Ratschek and J. Rokne. Interval methods. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 751–828. Kluwer Academic Publishers, The Netherlands, 1995.

[54] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods, part I: clustering methods. *Mathematical Programming*, 39: 27–56, 1987.

[55] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic global optimization methods, part II: multi level methods. *Mathematical Programming*, 39: 57–78, 1987.

[56] A.H.G. Rinnooy Kan and G.T. Timmer. Global optimization. In G.L. Nemhauser and oth., editors, *Handbooks in OR & MS*, volume 1, chapter IX, pages 631–662. Elsevier Science Publishers B.V. (North-Holland), 1989.

[57] H.E. Romeijn. *Global Optimization by Random Walk Sampling Methods*. Thesis Publishers, Amsterdam, The Netherlands, 1992.

[58] H.E. Romeijn. A general framework for approximate sampling with an application to generating points on the boundary of bounded convex regions. *Statist. Neerlandica*, 52(1): 42–59, 1998.

[59] H.E. Romeijn and R.L. Smith. Simulated annealing and adaptive search in global optimization. *Probability in the Engineering and Informational Sciences*, 8: 571–590, 1994.

[60] H.E. Romeijn and R.L. Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5(2): 101–126, 1994.

[61] R.Y. Rubinstein. *Simulation and Monte Carlo Method*. Wiley, New York, 1981.

[62] F. Schoen. Stochastic techniques for global optimization: a survey of recent advances. *Journal of Global Optimization*, 1: 207–228, 1991.

[63] P. Serafini. *Ottimizzazione*. Zanichelli, 2000.

[64] Ya.D. Sergeyev and V.A. Grishagin. Sequential and parallel algorithms for global optimization. *Optimization Methods and Software*, 3: 111–124, 1994.

[65] F.J. Solis and R.J.-B Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6: 19–30, 1981.

[66] C.P. Stephens and W. Baritompa. Global optimization requires global information. *JOTA*, 96(3): 575–588, 1998.

[67] R.G. Strongin. Numerical methods for multiextremal mathematical programming problemswith nonconvex constraints. In V.F. Demyanov and D. Pallaschke, editors, *Lecture Notes in Economics and Mathematical Systems*, volume 255, pages 278–282. Springer, Berlin, 1984.

[68] R.G. Strongin and Ya.D. Sergeyev. *Global Optimization with Non-Convex Constraints*. Kluwer Academic Publishers, 2000.

[69] A. Telcs. The art of random walks. In *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2006.

[70] A. Törn. A search-clustering approach to global optimization. In L.C.W. Dixon and G.P. Szegö, editors, *Towards Global Optimization 2*, pages 49–62. North Holland, Amsterdam, 1978.

[71] A. Törn and A. Žilinskas. Global optimization. In *Lecture Notes in Computer Science*, volume 350. Springer-Verlag, Berlino, 1989.

[72] H. Tuy. Convex analysis and global optimization. In *Nonconvex Optimization and its Applications*, volume 22. Kluwer Academic Publishers, Dordrecht, 1998.

[73] S.A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

[74] S.A. Vavasis. Black-box complexity of local minimization. *Siam J. Optimization*, 3(1): 60–80, 1993.

[75] S.A. Vavasis. Complexity issues in global optimization: a survey. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 27–41. Kluwer Academic Publishers, 1995.

[76] A. Žilinskas. Two algorithms for one-dimensional multimodal minimization. *Optimization*, 12: 53–63, 1981.

[77] Z.B. Zabinsky and R.L. Smith. Pure adaptive search in global optimization. *Math. Programming*, 53(3): 323–338, 1992.