Metodi ibridi di ottimizzazione: algoritmi metaeuristici seguiti da metodi basati sul gradiente

Corso: Metodi di Ottimizzazione Stocastici

Professori: Elisa Iacomini, Lorenzo Pareschi, Luca Saluzzi
Studentessa: Beatrice Laureti

Anno Accademico 2025/2026

Abstract

In questo progetto vengono studiati metodi ibridi di ottimizzazione che combinano algoritmi metaeuristici basati su popolazione di particelle con metodi basati sul gradiente.

Le strategie proposte vengono applicate a una nota funzione di test, Rastrigin, adatta a valutare la capacità degli algoritmi di gestire la presenza di numerosi minimi locali. L'obiettivo è quello di utilizzare questi algoritmi ibridi per minimizzare la funzione di test.

In particolare, vengono implementate e analizzate tre strategie ibride: Particle Swarm Optimization seguita da Adam, Particle Swarm Optimization seguita da Stochastic Gradient Descent e Consensus-Based Optimization seguita da Stochastic Gradient Descent.

Nei metodi proposti, la fase metaeuristica viene utilizzata come procedura di pre-inizializzazione, con l'obiettivo di individuare regioni favorevoli nello spazio delle soluzioni da cui avviare l'ottimizzazione locale.

L'analisi si concentra sul ruolo della metaeuristica nel fornire punti iniziali di qualità, sulla capacità di evitare minimi locali e sul raffinamento della soluzione con l'utilizzo dei metodi basati sul gradiente. In questo modo, si valuta la capacità dell'algoritmo ibrido di bilanciare l'esplorazione globale con l'ottimizzazione locale.

Introduzione

I metodi metaeuristici sono particolarmente efficaci nell'affrontare problemi di ottimizzazione non convessi caratterizzati dalla presenza di numerosi minimi locali. Essi riescono a esplorare abbastanza bene lo spazio di ricerca, in particolare, riescono ad individuare zone promettenti, riducendo quindi la dipendenza dall'inizializzazione. Tuttavia, una volta raggiunta una regione favorevole, la convergenza può risultare spesso lenta e poco accurata.

I metodi basati sul gradiente, invece, sfruttano informazioni locali della funzione obiettivo e consentono una convergenza rapida ed efficiente. Tuttavia, essi sono fortemente sensibili all'inizializzazione del punto e possono facilmente rimanere intrappolati in minimi locali, soprattutto in problemi ad alta dimensionalità.

Un approccio ibrido riesce a combinare i vantaggi dei metodi metaeuristici con quelli basati sul gradiente: viene utilizzato un metodo metaeuristico come fase di pre-inizializzazione per individuare una buona regione di partenza e un metodo basato sul gradiente per ottenere un raffinamento locale più accurato della soluzione. In questo modo si ottiene un compromesso efficace tra esplorazione globale e ottimizzazione locale.

L'approccio ibrido inverso non avrebbe senso, poiché i metodi basati sul gradiente sono fortemente dipendenti dal punto iniziale e tendono a convergere verso un minimo locale, limitando l'esplorazione. Inoltre, applicare successivamente una metaeuristica vanificherebbe il beneficio della convergenza locale già ottenuta e sarebbe inutilmente costosa computazionalmente.

Lo scopo di questo elaborato è analizzare l'efficacia di alcune strategie di ottimizzazione ibride, valutandone la capacità di minimizzare una funzione obiettivo.

Funzione di test: Rastrigin

Per valutare la capacità dei metodi ibridi di trovare il minimo globale di una funzione è stata scelta una nota funzione test: la funzione Rastrigin. Questa è non convessa e viene comunemente utilizzata per testare algoritmi che devono trovare il minimo di una funzione poichè presenta diversi minimi locali, ma un solo minimo globale che si trova in 0 con $f(0) = 0$. Più ci si allontana dall'origine e più i valori assunti dai minimi locali saranno alti. Rastrigin presenta una superficie altamente ondulata con numerosi minimi locali molto ampi, in cui è facile che l'algoritmo rimanga intrappolato.

$$f_{\mathrm{R}}(\mathbf{x}) = 10\,d + \sum_{i=1}^{d} \Big[x_i^2 - 10\cos(2\pi x_i) \Big], \quad \mathbf{x} \in \mathbb{R}^d$$

Algoritmi metaeuristici

Gli algoritmi metaeuristici cercano soluzioni accettabili a problemi complessi il cui obiettivo è quello di esplorare in modo efficace ed efficiente lo spazio di ricerca per determinare il minimo di una funzione obiettivo (o fitness). Tale ricerca avviene cercando di bilanciare due tecniche principali: l'**intensificazione**, che è l'esplorazione approfondita di alcune aree di ricerca che sembrano particolarmente promettenti e la **diversificazione**, che è l'esplorazione di nuove aree su cui intensificare successivamente la ricerca.

Tali metodi sostituiscono la valutazione del gradiente, che spesso può risultare costoso, non disponibile o ben definito, ad una combinazione di mosse casuali e deterministiche con strategie locali e globali al fine di effettuare una ricerca robusta della soluzione.

Questi algoritmi possono essere classificati in base al numero di soluzioni che utilizzano per determinare la soluzione finale:

- **Metodi a traiettoria:** il processo di ricerca è caratterizzato da una traiettoria nello spazio di ricerca e si basa sul miglioramento iterativo di una singola soluzione.
- **Metodi basati su popolazione:** il processo di ricerca gestisce un insieme di soluzioni ad ogni iterazione e può essere visto come l'evoluzione nel tempo di un insieme di punti nello spazio di ricerca.

In questo elaborato, ci concentreremo sui metodi basati su popolazione, in particolare su PSO e CBO, i quali verranno utilizzati come fase di pre-inizializzazione nei metodi ibridi.

Particle Swarm Optimization (PSO)

L'algoritmo PSO parte da una popolazione di soluzioni candidate, che sono rappresentate da delle particelle; esse modificano dinamicamente la propria posizione e velocità muovendosi lungo lo spazio di ricerca. La variazione della velocità di ogni particella è influenzata dalla miglior posizione ottenuta dalla particella stessa, detta *posizione migliore locale*, ma anche dalla miglior posizione ottenuta tra tutte le particelle nello spazio di ricerca, detta *posizione migliore collettiva*.

Algorithm 1 Particle Swarm Optimization (PSO)

for $i = 1$ to N **do**
  $(X_i^0, V_i^0) \leftarrow$ InitializeParticle()
end for
$Y_i^0 \leftarrow X_i^0$ ▷ Inizializzazione posizione migliore locale
$\bar{Y}^0 = \min\{f(Y_i^0)\}$ for $i = 1$ to N ▷ Inizializzazione posizione migliore globale
$n \leftarrow 0$
while termination criteria not met **do**
  for $i = 1$ to N **do** ▷ Per ogni particella
    $f_i^n = f(X_i^n)$ ▷ Valutazione della fitness
    if $f_i^n < f(Y_i^n)$ **then**
      $Y_i^n = X_i^n$
    end if
    if $f(Y_i^n) < f(\bar{Y}^n)$ **then**
      $\bar{Y}^n = Y_i^n$
    end if
  end for
  for $i = 1$ to N **do** ▷ Per ogni particella
    $V_i^{n+1} = wV_i^n + c_1 R_{1,i}^n (Y_i^n - X_i^n) + c_2 R_{2,i}^n (\bar{Y}^n - X_i^n)$ ▷ Aggiornamento velocità
    $X_i^{n+1} = X_i^n + V_i^{n+1}$ ▷ Aggiornamento posizione
  end for
  $n \leftarrow n + 1$
end while
return $\bar{Y}^n$

Funzionamento algoritmo

1. Si genera la popolazione iniziale di N particelle, data dall'insieme (X_i^0, V_i^0) con $i = 1, \dots N$, che è formato rispettivamente dalla posizione e dalla velocità della particella i-esima
2. Si valuta la fitness per ogni particella: $f_i^n = f(X_i^n)$ e si pone $Y_i^n = X_i^n$
3. Le particelle si muovono modificando opportunamente la propria posizione e velocità (X_i^{n+1}, V_i^{n+1})

4. Si valuta la fitness sulle nuove particelle: $f_i^{n+1} = f_i(X^{n+1})$
5. Per ogni particella se $f_i^{n+1} < f(Y_i^n)$ si registra la posizione migliore locale $Y_i^{n+1} = X_i^{n+1}$, altrimenti si pone $Y_i^{n+1} = Y_i^n$
6. La ricerca termina quando tutte le velocità V_i^{n+1} sono vicine allo zero, ovvero le particelle sono finite in un unico punto. In caso contrario si procede alla prossima iterazione e si ritorna al passo 3.

Posizione e velocità delle particelle

L'aggiornamento delle posizioni e velocità delle particelle avviene secondo le seguenti regole:

$$X_i^{n+1} = X_i^n + V_i^{n+1},$$

$$V_i^{n+1} = wV_i^n + c_1 R_{1,i}^n (Y_i^n - X_i^n) + c_2 R_{2,i}^n (\bar{Y}^n - X_i^n),$$

dove $c_1, c_2 \in \mathbb{R}$ sono i *coefficienti di accelerazione*, w è il *peso inerziale*, Y_i^n è la posizione migliore locale trovata dalla particella i fino a quella iterazione e $\bar{Y}_i^n = \text{argmin}\{f(Y_i^n), \dots, f(Y_N^n)\}$ è la miglior posizione globale trovata tra tutte le particelle fino a quella iterazione. $R_{1,i}^n$ e $R_{2,i}^n$ sono due matrici diagonali di dimensione d aventi sulla diagonale numeri casuali distribuiti uniformemente in $[0, 1]$, che sono generati ad ogni iterazione per ciascuna particella.

Consensus-Based Optimization (CBO)

L'algoritmo CBO è ispirato alle dinamiche di auto-organizzazione e allineamento collettivo; infatti, utilizza una popolazione di particelle che esplora lo spazio delle soluzioni combinando un meccanismo di allineamento verso una stima condivisa del minimo globale con una componente di esplorazione stocastica.

Algorithm 2 Consensus-Based Optimization (CBO)

for $i = 1$ to N **do**
  $X_i^0 \leftarrow$ InitializeParticle() ▷ Inizializzazione particelle
end for
$n \leftarrow 0$
while termination criteria not met **do**
  for $i = 1$ to N **do** ▷ Calcolo fitness per ogni particella
    $f_i^n = f(X_i^n)$
  end for
  $\bar{X}_\alpha^n \leftarrow \dfrac{\sum_{i=1}^N X_i^n \, e^{-\alpha f(X_i^n)}}{\sum_{i=1}^N e^{-\alpha f(X_i^n)}}$ ▷ Calcolo punto di consenso
  for $i = 1$ to N **do**
    $\xi_i^n \leftarrow$ SampleNormalVector()
    $X_i^{n+1} = X_i^n - \Delta t \, \lambda (X_i^n - \bar{X}_\alpha^n) + \sqrt{\Delta t} \, \sigma \, D(X_i^n - \bar{X}_\alpha^n) \, \xi_i^n$ ▷ Aggiornamento posizione
  end for
  $n \leftarrow n + 1$
end while
return $\bar{X}_\alpha^n$ ▷ Stima del minimo globale

L'algoritmo CBO può essere descritto come un'evoluzione semplice di N particelle con posizioni $X_i^n \in \mathbb{R}^d$ al tempo n secondo la dinamica discreta:

$$X_i^{n+1} = X_i^n - \Delta t \, \lambda \big(X_i^n - \bar{X}_\alpha^n\big) \; + \; \sqrt{\Delta t} \, \sigma \, D\big(X_i^n - \bar{X}_\alpha^n\big) \, \xi_i^n,$$

dove il primo termine rappresenta il contributo di *allineamento* verso il punto di consenso, mentre il secondo descrive il termine di *esplorazione* stocastica.

Le variabili casuali ξ_i^n sono indipendenti e distribuite secondo una legge normale standard $N(0, 1)$

La matrice di diffusione $D(\cdot)$ può essere scelta in diverse forme, tra cui:

- esplorazione isotropa: $D(X) = \|X_i\|_2 \, I_d$; il rumore ha la stessa intensità in tutte le direzioni dello spazio.
- esplorazione anisotropa: $D(X_i) = \text{diag}((X_i)_1, (X_i)_2, \dots, (X_i)_d)$; il rumore dipende dalla direzione o dalla singola coordinata.

Il punto di consenso (o stima del minimo globale) è definito come:

$$\bar{X}_\alpha^n = \frac{\sum_{i=1}^N X_i^n \, \omega_\alpha^f(X_i^n)}{\sum_{i=1}^N \omega_\alpha^f(X_i^n)},$$

dove i pesi sono dati dalla distribuzione di Gibbs associata a f, $\omega_\alpha^f(x) = \exp\big(-\alpha f(x)\big)$.

Per $\alpha = 0$, il punto di consenso coincide con la media aritmetica della popolazione, mentre per valori grandi di α, in virtù del principio di Laplace, si ha l'approssimazione

$$\bar{X}_\alpha^n \approx \arg\min\{f(X_1^n), \dots, f(X_N^n)\}.$$

I parametri $\lambda > 0$ e $\sigma \geq 0$ rappresentano rispettivamente il *parametro di consenso* e il *parametro di esplorazione*.

Algoritmi basati sul gradiente

I metodi basati sul gradiente sfruttano informazioni locali della funzione obiettivo per guidare l'aggiornamento dei parametri nella direzione di massima discesa. Essi risultano particolarmente efficienti in prossimità di un minimo, garantendo una rapida convergenza. Esistono varianti stocastiche che introducono una componente di rumore nel gradiente, riducendo il costo computazionale e favorendo una maggiore capacità esplorativa.

Tali metodi risultano particolarmente efficaci per il raffinamento locale di una soluzione. Nel seguito, introduciamo due noti metodi basati sul gradiente che utilizzeremo nell'implementazione degli algoritmi ibridi: SGD e Adam.

Stochastic Gradient Descent (SGD)

Nel contesto dell'apprendimento supervisionato si considera un insieme di dati $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$,

dove $x_i \in \mathbb{R}^d$ rappresenta l'input e $y_i \in \mathbb{R}^m$ l'output associato. L'obiettivo è determinare i parametri θ di un modello parametrico $f(x;\theta)$ minimizzando la perdita empirica

$$\min_\theta F(\theta) = \frac{1}{N} \sum_{i=1}^N l\big(f(x_i;\theta), y_i\big).$$

Il gradiente della funzione obiettivo è dato da:

$$\nabla_\theta F(\theta) = \frac{1}{N} \sum_{i=1}^N \nabla_\theta l_i(\theta), \qquad \text{dove} \quad l_i(\theta) := l\big(f(x_i;\theta), y_i\big).$$

La discesa del gradiente standard aggiorna i parametri muovendosi nella direzione opposta al gradiente della funzione obiettivo, poichè il gradiente rappresenta la direzione di massima crescita:

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta F(\theta_k),$$

dove η è il learning rate o anche detto passo, e stabilisce quanto ci si sposta nella direzione del gradiente ad ogni iterazione.

La discesa del gradiente standard richiede un costo computazionale per iterazione proporzionale a N e può restare intrappolata in minimi locali. Per ovviare a questi limiti, si utilizza il gradiente stocastico, che approssima il gradiente completo utilizzando il gradiente di un singolo campione $i_k \sim \mathcal{U}\{1, \dots, N\}$:

$$g(\theta_k, i_k) = \nabla_\theta l_{i_k}(\theta_k) \ , \qquad \theta_{k+1} = \theta_k - \eta_k g(\theta_k, i_k).$$

Una variante dello SGD consiste nell'utilizzo dei mini-batch: invece di aggiornare i parametri usando un singolo campione per iterazione, si considerano dei sottoinsiemi del dataset di dimensione B e ad ogni iterazione, si calcola il gradiente medio sul batch, riducendo così la varianza del gradiente.

Random Coordinate Descent

Nel nostro caso non stiamo trattando un problema di apprendimento supervisionato e non disponiamo di un insieme di campioni, ma vogliamo semplicemente minimizzare una funzione obiettivo f.

Per questo motivo, utilizziamo l'algoritmo Random Coordinate Descent (RCD), interpretabile come una forma di SGD mini-batch in cui la stocasticità agisce sulle coordinate anziché sui dati. Il punto viene aggiornato usando il gradiente solo rispetto a un sottoinsieme casuale di componenti, mentre le altre restano invariate. In questo modo, otteniamo una riduzione del costo computazionale per iterazione e introduciamo un effetto di rumore che favorisce l'esplorazione, similmente a quanto avviene nello SGD.

Nell'implementazione proposta in seguito, ogni riferimento a SGD va inteso come riferito alla sua variante Random Coordinate Descent.

Algorithm 3 Random Coordinate Descent (RCD-SGD)

$x^0 \in \mathbb{R}^d$, $k = 0$ ▷ Inizializzazione punto
while termination criteria not met **do**
  $\eta_k \leftarrow \eta_0/k$ ▷ Passo decrescente
  Draw a subset $D_k \subset \{1, \dots, d\}$ uniformly at random ▷ Scelta coordinate da aggiornare
  for each $j \in D_k$ **do**
    $x_j^{k+1} \leftarrow x_j^k - \eta_k \, \partial_j f(x^k)$ ▷ Aggiornamento coordinate
  end for
  $k \leftarrow k + 1$
end while

Adaptive Moment Estimation (Adam)

Si consideri il problema di minimizzazione $\min_{\theta \in \mathbb{R}^d} f(\theta)$, in cui il gradiente non è calcolato esattamente ma stimato in modo stocastico. Nel metodo SGD con momento, l'aggiornamento è dato da:

$$m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k, \qquad \theta_{k+1} = \theta_k - \eta_k m_{k+1},$$

dove g_k è uno stimatore stocastico del gradiente. SGD con momento tiene conto anche delle pendenze passate, riuscendo così, ad attenuare le oscillazioni attorno alla soluzione e ad accelerare la convergenza lungo direzioni 'giuste'. Tuttavia, esso utilizza un unico learning rate per tutte le componenti che può risultare inefficace, soprattutto in presenza di gradienti sparsi.

I metodi adattivi (come Adagrad e RMSProp) invece, introducono una scalatura componente per componente, sfruttando l'informazione storica sui gradienti per adattare il passo di aggiornamento.

Adam combina i vantaggi dello SGD con momento, che fornisce una stima della direzione media di discesa, e di RMSProp, che introduce una scalatura adattiva del passo basata sui quadrati dei gradienti.

L'obiettivo è ottenere un metodo semplice, con costo per iterazione simile a SGD, ma capace di adattare automaticamente il learning rate per ciascuna coordinata.

Algorithm 4 Adam con Random Coordinate Descent

$x^0 \in \mathbb{R}^d$, $m^0 = 0$, $v^0 = 0$, $k = 0$ ▷ Inizializzazione
while termination criteria not met **do**
  Draw a subset $D_k \subset \{1, \dots, d\}$ uniformly at random
  for each $j \in D_k$ **do**
    $g_{k,j} \leftarrow \partial_j f(x^k)$ ▷ Gradiente stocastico per coordinate
    $m_{k+1,j} \leftarrow \beta_1 m_{k,j} + (1 - \beta_1) g_{k,j}$ ▷ Momento primo
    $v_{k+1,j} \leftarrow \beta_2 v_{k,j} + (1 - \beta_2) g_{k,j}^2$ ▷ Momento secondo
    $\hat{m}_{k+1,j} \leftarrow \dfrac{m_{k+1,j}}{1 - \beta_1^{k+1}}$ ▷ Correzione bias momento primo
    $\hat{v}_{k+1,j} \leftarrow \dfrac{v_{k+1,j}}{1 - \beta_2^{k+1}}$ ▷ Correzione bias momento secondo
    $x_j^{k+1} \leftarrow x_j^k - \eta \dfrac{\hat{m}_{k+1,j}}{\sqrt{\hat{v}_{k+1,j} + \varepsilon}}$ ▷ Aggiornamento coordinate
  end for
  $k \leftarrow k + 1$
end while

Adam calcola:

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k, \quad v_k = \beta_2 v_{k-1} + (1 - \beta_2) g_k^{\odot 2}, \quad \hat{m}_k = \frac{m_k}{1 - \beta_1^k}, \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^k},$$

dove $g_k^{\odot 2}$ denota il quadrato componente per componente del gradiente g_k stocastico (nel nostro caso è random coordinate descent), m_k è una stima del momento primo, v_k è la stima del momento secondo, $\beta 1, \beta 2 \in [0, 1)$ sono i coefficienti di decadimento che regolano rispettivamente la media mobile dei gradienti (momento primo) e la media mobile dei gradienti al quadrato (momento secondo).

Poiché m_0 e v_0 sono inizializzati a zero, le stime risultano inizialmente biasate; Adam introduce quindi una correzione del bias $\hat{m}_k, \hat{v}_k$.

L'aggiornamento dei parametri è infine dato da: $\theta_{k+1} = \theta_k - \eta \dfrac{\hat{m}_k}{\sqrt{\hat{v}_k + \varepsilon}}$,

dove il rapporto è inteso componente per componente e ε garantisce stabilità numerica.

Implementazione metodi

Nel seguito viene presentata la scelta dei parametri relativi ai diversi metodi al fine di garantire la riproducibilità degli stessi e sui quali si basa la discussione dei risultati ottenuti fornita successivamente.

Poiché il comportamento dell'algoritmo dipende fortemente dai parametri selezionati, è stato necessario adattarli alla dimensione del problema e alla funzione da minimizzare. In particolare, l'algoritmo viene studiato per alcune dimensioni rappresentative (2, 10, 50), riportando per ciascun caso una possibile scelta adeguata dei parametri. Si può osservare come delle configurazioni scelte per una certa dimensione non risultino ottimali per altre.

L'individuazione di tali parametri è avvenuta attraverso numerosi test empirici; tuttavia, data la complessità del problema e l'elevato numero di combinazioni possibili, non si esclude l'esistenza di una scelta migliore per i problemi considerati.

Per garantire un confronto equo e immediato tra PSO e CBO, sono stati fissati gli stessi valori per il numero di particelle, per il numero massimo di iterazioni e per il criterio di arresto basato sull'assenza di miglioramenti. Per lo stesso motivo, per SGD e Adam, sono stati fissati gli stessi valori per il learning rate (per SGD è inteso come η_0), per il numero massimo di iterazioni e per il criterio di arresto basato sull'assenza di miglioramenti.

	PSO e CBO			**SGD e Adam**		
N	n	n_{it}	$n_{it_{sm}}$	η	n_{it}	$n_{it_{sm}}$
2	50	100	40	$10^{-4} - 10^{-2}$	200	50
10	300	200	40	$10^{-4} - 10^{-2}$	300	50
50	3000	500	40	$10^{-4} - 10^{-2}$	500	50

All'aumentare della dimensione del problema è stato necessario incrementare il numero di particelle n per i metodi basati su popolazione, il numero di iterazioni n_{it} e η per i metodi basati sul gradiente, al fine di favorire una maggiore esplorazione dello spazio di ricerca ed evitare convergenza prematura.

Criteri di terminazione

Tutti i metodi considerati condividono i seguenti criteri di arresto:

- Raggiungimento del numero massimo di iterazioni n_{it};
- Superamento del numero massimo di iterazioni consecutive senza miglioramento $n_{it_{sm}}$, che indica stagnazione nella ricerca.

I metodi metaeuristici hanno un ulteriore criterio di terminazione legato alla dinamica collettiva delle particelle, esso prevede il collasso della popolazione:

- PSO: convergenza delle velocità, quando la norma massima delle velocità delle particelle scende al di sotto di una soglia prefissata.
- CBO: la dispersione massima delle particelle rispetto al punto di consenso scende sotto una certa soglia. Indica che le particelle hanno raggiunto un consenso.

Implementazione PSO

- w: valori più elevati favoriscono l'esplorazione, mentre valori più bassi incentivano la convergenza verso soluzioni promettenti.

  Nell'implementazione dell'algoritmo si è scelto di utilizzare un coefficiente di inerzia costante, al fine di isolare l'effetto degli altri parametri e semplificare il confronto tra i metodi.
- c_1: regola l'influenza della migliore posizione personale, aumentando l'esplorazione individuale delle particelle.
- c_2: regola l'attrazione verso la migliore soluzione globale, favorendo l'intensificazione della ricerca.
- n: numero di particelle della popolazione, che influisce sulla copertura dello spazio di ricerca.

Sono stati scelti: $w = 0.4 - 1,\ c_1 = c_2 = 1.5$.

Implementazione CBO

In questo paragrafo seguono alcune scelte implementative dell'algoritmo; in particolare, per quanto riguarda la tipologia d'esplorazione, è stata scelta quella anisotropa poichè il rumore agisce in modo diverso sulle varie coordinate. Questa scelta permette di mantenere una buona capacità esplorativa senza avere eccessive oscillazioni nelle fasi finali dell'algoritmo, soprattutto per problemi non convessi.

- α: intensità di selezione; valori più elevati concentrano il consenso sulle particelle migliori, al contrario, valori minori favoriscono un comportamento più esplorativo.
- λ: parametro di consenso, controlla la velocità di allineamento delle particelle.
- σ: intensità del rumore, regola l'intensità del rumore stocastico.
- Δt: passo temporale della dinamica discreta.
- n: numero di particelle della popolazione, che influisce sulla copertura dello spazio di ricerca.

Sono stati scelti: $\alpha = 100,\ \lambda = 0.2 - 0.5,\ \sigma = 1.0 - 2.0,\ \Delta t = 0.1$.

Implementazione RCD-SGD

- η_0: learning rate iniziale; determina l'ampiezza del primo passo di discesa e viene successivamente ridotto secondo una legge di tipo Robbins–Monro.
- η_k: learning rate all'iterazione k, definito come $\eta_k = \eta_0/k$.
- d: numero di coordinate aggiornate a ogni iterazione, selezionate uniformemente a caso, in questo caso $d = \lceil \dim/5 \rceil$.

Implementazione Adam

- η: learning rate di base, controlla l'ampiezza dell'aggiornamento dei parametri ed è adattivamente scalato, per ciascuna coordinata, tramite le stime dei momenti primo e secondo.
- β_1: coefficiente di decadimento del momento primo, regola quanto i gradienti passati influenzano la stima della direzione media di discesa.
- β_2: coefficiente di decadimento del momento secondo, controlla la media mobile dei quadrati dei gradienti, determinando la scalatura adattiva del passo per componente.
- ε: parametro di stabilità numerica, introdotto per evitare divisioni per valori troppo piccoli.
- d: numero di coordinate aggiornate a ogni iterazione, selezionate uniformemente a caso, in questo caso $d = \lceil \dim/5 \rceil$.

Sono stati scelti: $\beta 1 = 0.5 - 0.9,\ \beta 2 = 0.8 - 0.999,\ \varepsilon = 10^{-8}$.

Risultati: confronto tra metodi e parametri relativi ad essi

Analisi degli algoritmi al variare di alcuni parametri chiave

Nel seguito vengono presentati alcuni esempi di esecuzione degli algoritmi, ottenuti variando opportunamente alcuni parametri chiave, riferiti a tre dimensioni rappresentative del problema (figure 1-9). Poiché il numero complessivo dei parametri è elevato, la selezione delle configurazioni mostrate è il risultato di numerosi test preliminari, finalizzati a individuare scelte ragionevoli e significative. In particolare, per ciascun algoritmo sono stati fatti variare soltanto uno o due parametri principali, considerandone due o tre valori ciascuno, così da mantenere i grafici di confronto leggibili ed evitare un'eccessiva complessità dovuta alla numerosità dei parametri. In particolare per PSO, si è deciso di variare w, per CBO λ e σ, per SGD il learning rate η e per Adam β_1 e β_2.

In questo paragrafo i metodi ibridi vengono analizzati separatamente, esaminandone il comportamento al variare dei parametri chiave selezionati. Il fine è quello di individuare le configurazioni più efficaci per ciascun algoritmo ibrido e riuscire successivamente a confrontare congiuntamente i tre metodi.

Per rendere il confronto grafico più chiaro e omogeneo, in questa fase è stato adottato come unico criterio di arresto il raggiungimento del numero massimo di iterazioni; gli altri criteri di terminazione vengono utilizzati nel paragrafo successivo, dedicato al confronto congiunto dei tre algoritmi ibridi (figure 10-12).

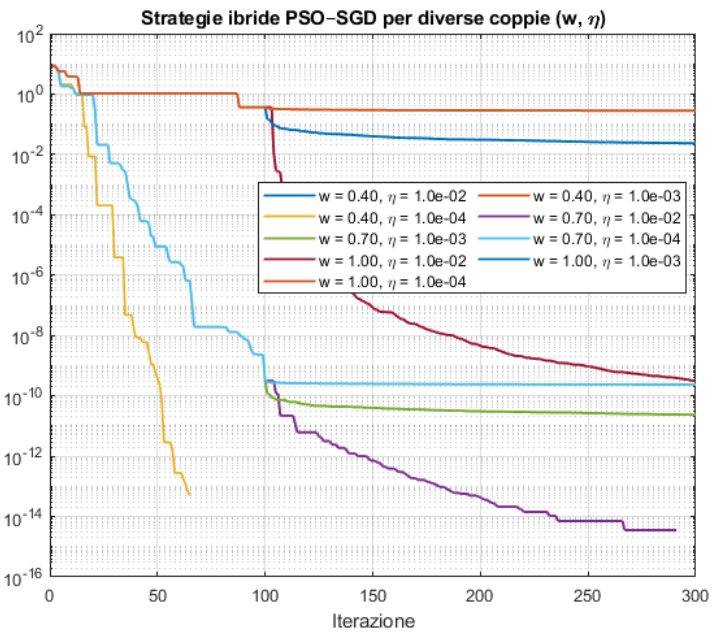

Figure 1: N=2: PSO-SGD al variare dei parametri chiave w e η.

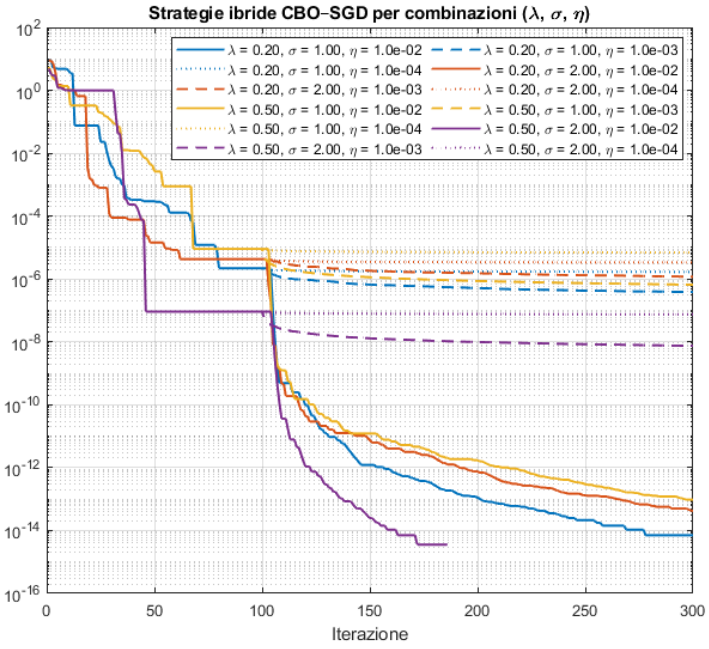

Figure 2: N=2: CBO-SGD al variare dei parametri chiave λ, σ e η.

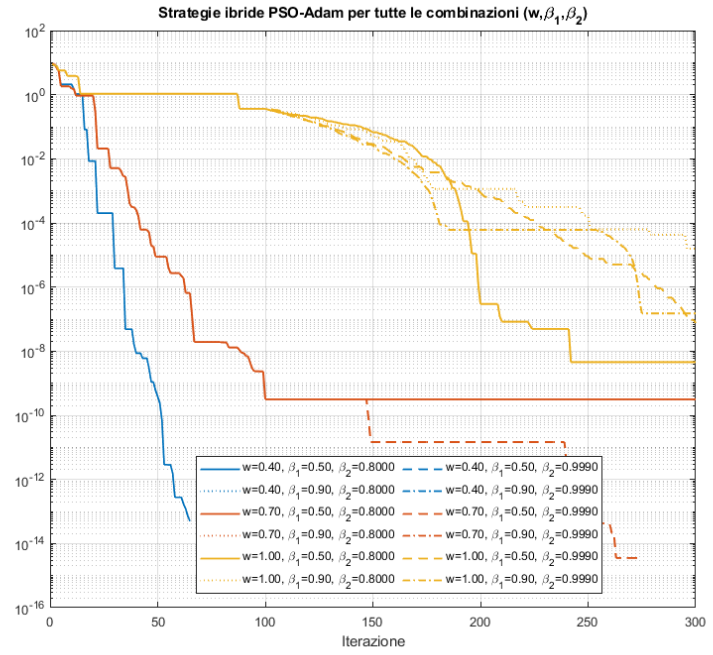

Figure 3: N=2: PSO-Adam al variare dei parametri chiave w, β_1 e β_2

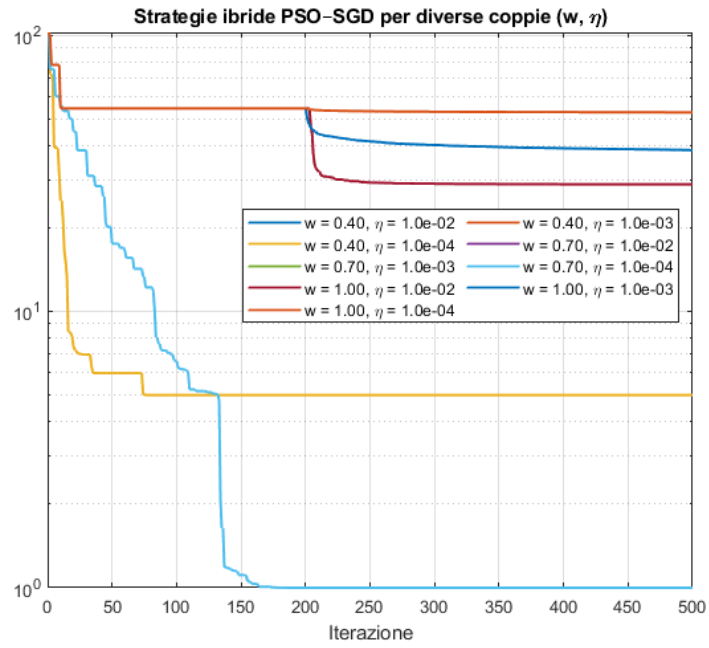

Figure 4: N=10: PSO-SGD al variare dei parametri chiave w e η.

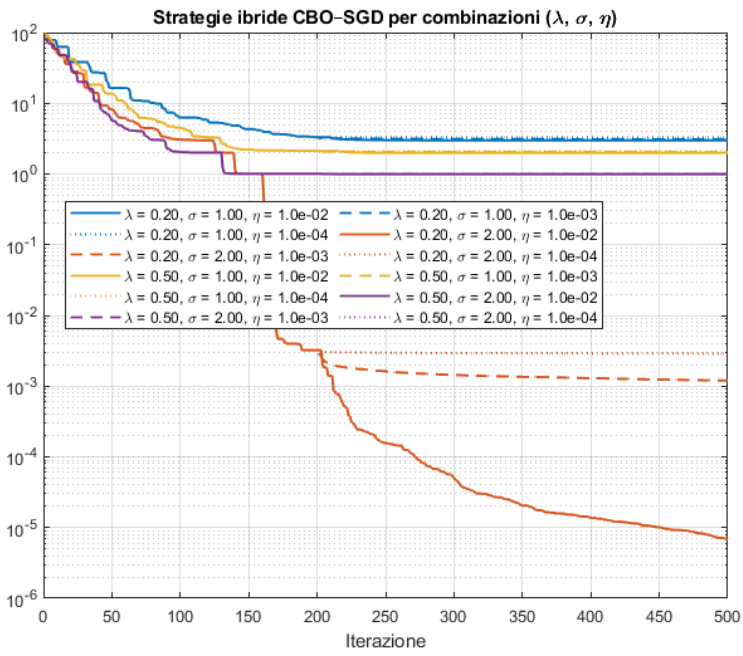

Figure 5: N=10: CBO-SGD al variare dei parametri chiave λ, σ e η.

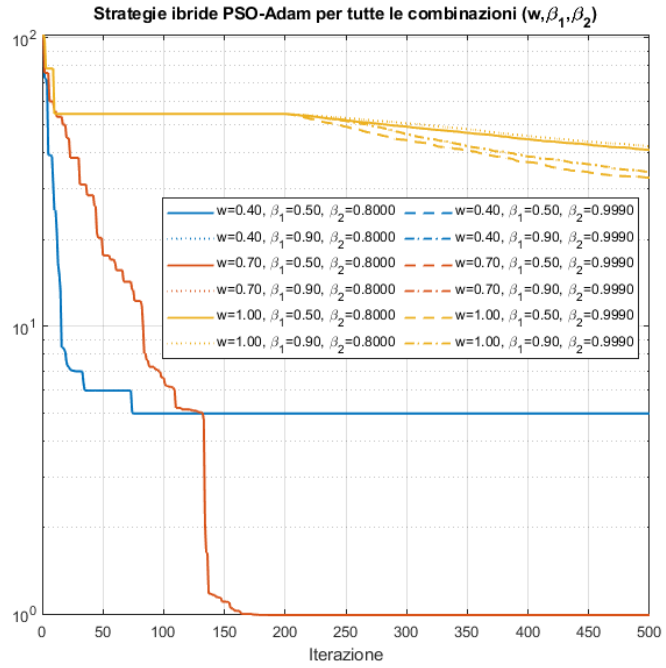

Figure 6: N=10: PSO-Adam al variare dei parametri chiave w, β_1 e β_2

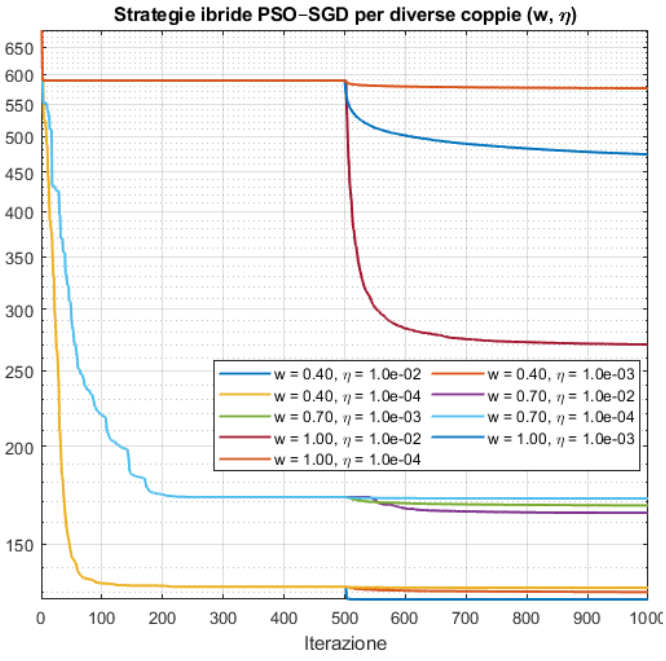

Figure 7: N=50: PSO-SGD al variare dei parametri chiave w e η.

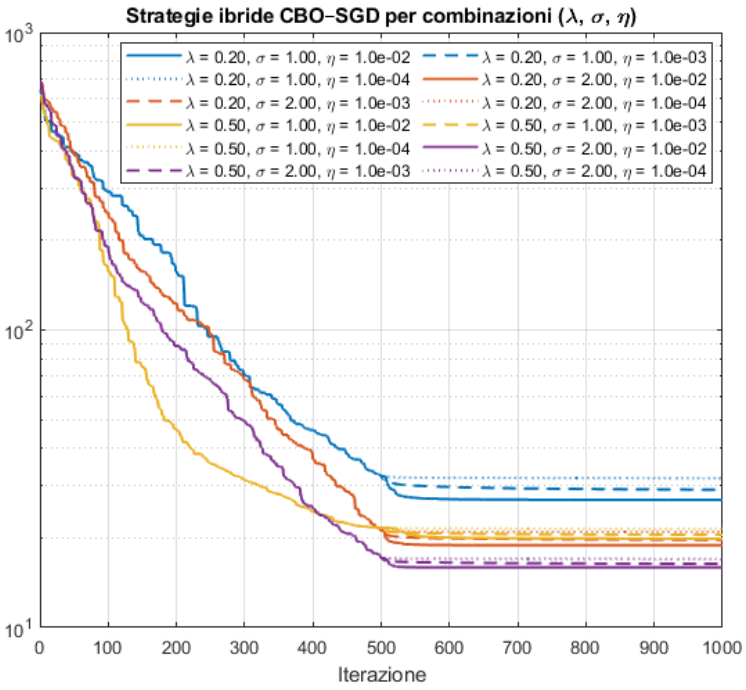

Figure 8: N=50: CBO-SGD al variare dei parametri chiave λ, σ e η.

Dall'analisi delle figure 1–9 emergono alcune indicazioni sul ruolo dei parametri chiave per ciascun metodo.

Per quanto riguarda PSO, valori per il coefficiente di inerzia compresi tra $w = 0.4$ e $w = 0.7$ risultano complessivamente più efficaci rispetto a inerzie più elevate, che tendono a penalizzare la convergenza favorendo l'esplorazione. Nel caso di CBO, valori più alti di σ favoriscono una maggiore esplorazione dello spazio di ricerca e si rivelano generalmente preferibili; per il parametro di consenso λ, valori compresi tra 0.2 e 0.5, offrono un buon compromesso tra esplorazione e consenso delle particelle. Per SGD, il learning rate $\eta = 10^{-2}$ si dimostra il più efficace per tutte le dimensioni considerate, in quanto consente una dinamica più esplorativa rispetto a valori più piccoli come 10^{-3} o 10^{-4}. Per Adam i risultati migliori si ottengono per valori elevati di β_1 e β_2, con $\beta_1 < \beta_2$, configurazione che garantisce una media più stabile dei gradienti e una migliore scalatura adattiva.

Nel caso $N = 2$ (figure 1-3), PSO è in grado di individuare rapidamente il minimo globale già nelle prime iterazioni, raggiungendo valori prossimi alla precisione di macchina. In questo contesto, l'applicazione successiva di SGD o Adam risulta superflua per il perfezionamento della soluzione. CBO, per tutte le configurazioni provate, riesce anch'esso a convergere verso il minimo globale. Tuttavia, in questo caso, l'applicazione di SGD con learning rate adeguato, consente un raffinamento significativo della soluzione, migliorandone ulteriormente l'accuratezza.

Per $N = 10$ (figure 4-6), PSO mostra una capacità esplorativa più limitata e tende a convergere verso un minimo locale, sebbene relativamente vicino a quello globale. In tale situazione, né SGD né Adam riescono a migliorare sensibilmente la soluzione, dal momento che il processo di raffinamento è già stato in larga parte effettuato da PSO. Al contrario, CBO si dimostra più efficace nell'esplorazione dello spazio e riesce a raggiungere il minimo globale. l'applicazione successiva di SGD dopo CBO, con un learning rate adeguato, consente in questo caso un buon raffinamento della soluzione.

Nel caso ad alta dimensionalità $N = 50$ (figure 7-9), nonostante l'aumento del numero di particelle e del numero massimo di iterazioni, né PSO né CBO riescono a raggiungere il minimo globale. Entrambi gli algoritmi rimangono intrappolati in minimi locali relativamente lontani, con valori della funzione obiettivo dell'ordine di $10^1 - 10^2$. In questa situazione, SGD e Adam non riescono a sfuggire dal minimo locale, limitandosi a raffinare la soluzione trovata senza avvicinarsi al minimo globale. Sebbene non riportato nei grafici, per dimensioni ancora più grandi, questo comportamento risulta ulteriormente accentuato, evidenziando le crescenti difficoltà legate all'aumento della dimensionalità del problema.

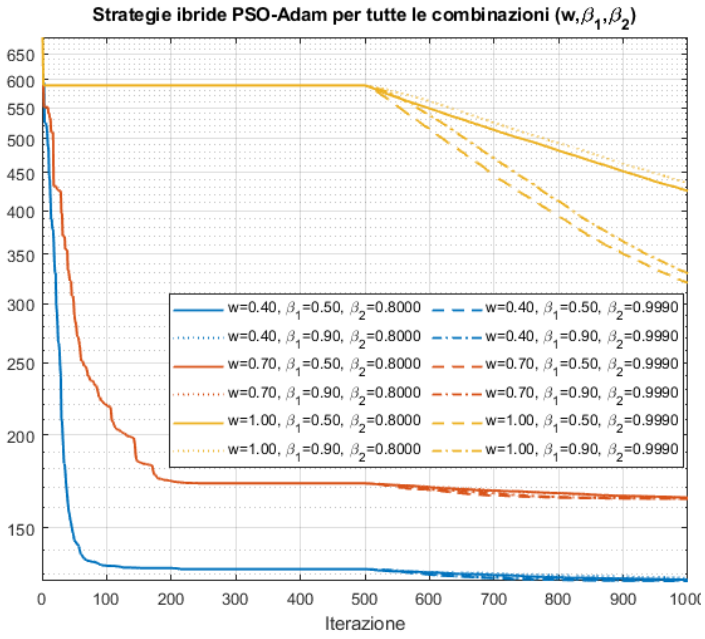

Figure 9: N=50: PSO-Adam al variare dei parametri chiave w, β_1 e β_2

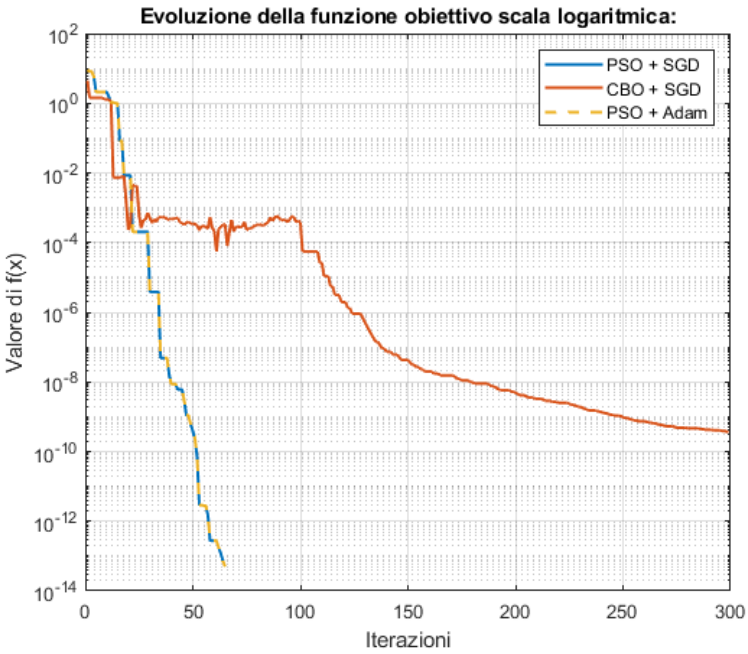

Figure 10: N=2: Confronto tra i metodi ibridi con parametri $w = 0.4, \lambda = 0.5, \sigma = 2, \eta = 10^{-2}, \beta_1 = 0.9, \beta_2 = 0.999$.

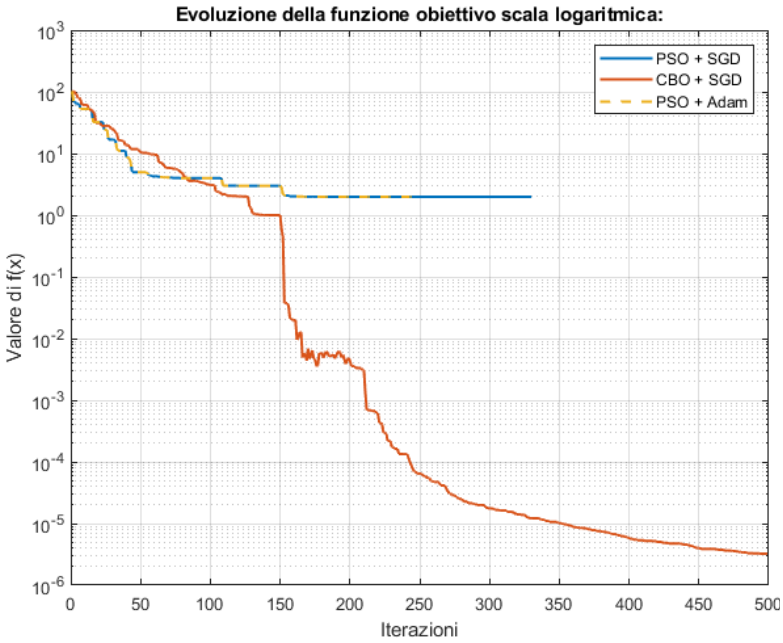

Figure 11: N=10: Confronto tra i metodi ibridi con parametri $w = 0.7, \lambda = 0.2, \sigma = 2, \eta = 10^{-2}, \beta_1 = 0.9, \beta_2 = 0.999$.

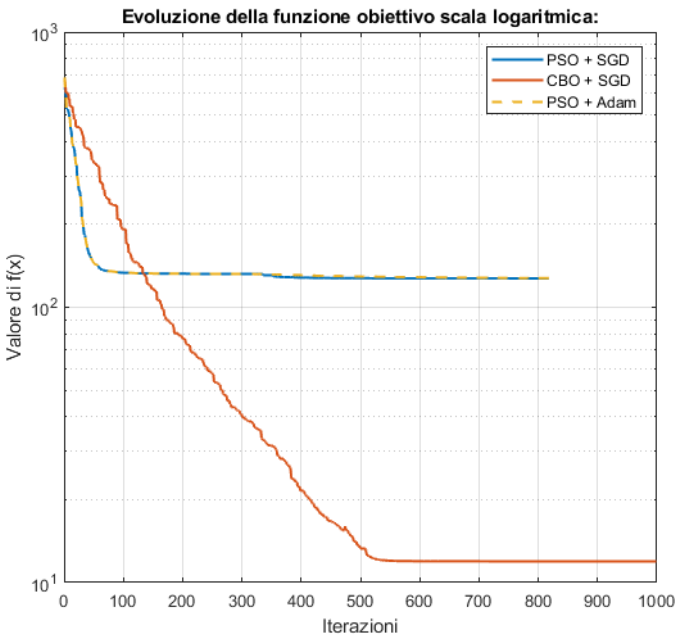

Figure 12: N=50: Confronto tra i metodi ibridi con parametri $w = 0.4, \lambda = 0.5, \sigma = 2, \eta = 10^{-2}, \beta_1 = 0.9, \beta_2 = 0.999$.

Analisi comparativa degli algoritmi ibridi

In questo paragrafo vengono confrontati i tre algoritmi ibridi (figure 10–12), utilizzando per ciascun metodo le configurazioni dei parametri che sono risultate più efficaci dall'analisi preliminare al variare dei parametri chiave. I parametri che in precedenza erano stati mantenuti fissi vengono lasciati invariati; inoltre vengono reintrodotti tutti i criteri di arresto esposti in precedenza.

Per quanto riguarda la fase metaeuristica, nonostante il costo computazionale elevato, essa si rivela fondamentale per l'esplorazione dello spazio delle soluzioni. Sia PSO che CBO hanno dimostrato di svolgere un ruolo cruciale nell'individuare regioni promettenti, soprattutto in dimensioni basse e intermedie del problema. In dimensione $N = 2$, PSO risulta più rapido di CBO nell'individuare la zona del minimo globale, mostrando una convergenza molto veloce già nelle prime iterazioni. Tuttavia, al crescere della dimensione, CBO si dimostra sensibilmente più esplorativo: per $N = 10$, PSO riesce ad avvicinarsi al minimo globale ma rimane intrappolato in un minimo locale, mentre CBO riesce a entrare nel bacino del minimo globale. Per $N = 50$, nessuno dei due metodi riesce a raggiungere il minimo globale. Infatti, entrambi i metodi rimangono intrappolati in minimi locali lontani (dell'ordine di $10^1 - 10^2$), sebbene CBO fornisca soluzioni migliori rispetto a PSO. Nel complesso, i metodi metaeuristici svolgono la parte predominante del lavoro esplorativo, ma sulla funzione di Rastrigin mostrano evidenti limiti al crescere della dimensionalità.

Per quanto riguarda i metodi basati sul gradiente, il loro contributo risulta fortemente dipendente dalla qualità della soluzione fornita dalla fase metaeuristica. Nel caso $N = 2$, SGD e Adam consentono un raffinamento efficace della soluzione, migliorando ulteriormente l'accuratezza, arrivando a raggiungere la precisione macchina. Per $N = 10$, SGD risulta particolarmente efficace nel perfezionare la soluzione dopo CBO, poiché quest'ultimo riesce a individuare il minimo globale. Al contrario, quando PSO converge a un minimo locale, né SGD né Adam riescono a fuoriuscirne, limitandosi a un leggero raffinamento della soluzione locale. Nel caso $N = 50$, il contributo dei metodi basati sul gradiente diventa marginale: SGD e Adam rimangono intrappolati nel minimo locale individuato nella fase metaeuristica, senza riuscire ad avvicinarsi al minimo globale. Questo evidenzia come i metodi basati sul gradiente siano efficaci solo quando la fase metaeuristica riesce a fornire una soluzione sufficientemente vicina al bacino del minimo globale; in caso contrario, la loro capacità esplorativa risulta insufficiente poichè tendono a rimanere intrappolati nei minimi locali della funzione, senza riuscire a raggiungere il bacino del minimo globale. In particolare, per la funzione Rastrigin, tali metodi non riescono a compensare una fase metaeuristica inefficace soprattutto in alta dimensione.

Dunque, dai risultati ottenuti non emergono casi in cui la fase metaeuristica peggiori la qualità finale della soluzione; essa risulta sempre necessaria per fornire una buona inizializzazione ai metodi basati sul gradiente, che mostrano una grande dipendenza dal punto iniziale.

Per dimensioni $N \geq 10$, CBO-SGD risulta più efficace rispetto alle combinazioni basate su PSO, grazie alla maggiore capacità esplorativa di CBO, che diventa essenziale all'aumentare della dimensione del problema. Al contrario, nel caso $N = 2$, PSO-SGD e PSO-Adam risultano più performanti rispetto a CBO-SGD, evidenziando come CBO tenda in questo caso a mantenere una dinamica esplorativa anche dopo aver individuato il minimo globale.

Nel confronto complessivo tra gli algoritmi ibridi, PSO-SGD e PSO-Adam hanno prestazioni molto simili, infatti, le soluzioni finali risultano praticamente equivalenti. Tuttavia, nel contesto della funzione Rastrigin, SGD risulta complessivamente più adatto rispetto ad Adam. Questo comportamento è dovuto al fatto che SGD ha una maggiore capacità esplorativa, che risulta vantaggiosa nel caso della Rastrigin. Adam introduce una stima adattiva dei momenti di primo e secondo ordine del gradiente, tendendo a stabilizzare eccessivamente gli aggiornamenti, favorendo una convergenza più rapida verso il minimo locale. In presenza di numerosi e ampi minimi locali, come nel caso della funzione Rastrigin, questa stabilità porta a una minore capacità di esplorazione e dunque, a una ridotta possibilità di miglioramento della soluzione. Inoltre, SGD risulta meno costoso a livello computazionale, rendendolo preferibile ad Adam nella funzione test considerata.

Conclusioni

L'analisi svolta mette in evidenza il ruolo centrale della fase metaeuristica all'interno degli algoritmi ibridi considerati: PSO e CBO si confermano algoritmi capaci di esplorare lo spazio delle soluzioni e di individuare regioni iniziali promettenti in dimensioni relativamente basse. In particolare, CBO mostra una maggiore capacità esplorativa rispetto a PSO, soprattutto al crescere della dimensione del problema.

I metodi basati sul gradiente, invece, svolgono un ruolo di raffinamento locale e la loro efficacia è fortemente condizionata dalla qualità della soluzione fornita dall'algoritmo metaeuristico. Infatti, in assenza di una buona inizializzazione, né SGD né Adam riescono a raggiungere il bacino del minimo globale, evidenziando una limitata capacità esplorativa. Dai risultati non emergono casi in cui la fase metaeuristica peggiori il processo di ottimizzazione; al contrario, essa risulta sempre necessaria e mai penalizzante.

Nel confronto tra i metodi ibridi, CBO-SGD si è dimostrato la combinazione migliore per dimensioni medio-alte, grazie alla maggiore esplorazione globale dovuta a CBO. PSO-SGD e PSO-Adam risultano invece più competitivi in bassa dimensione.

Nel complesso, SGD si rivela preferibile ad Adam per la funzione di Rastrigin, sia per la maggiore capacità esplorativa sia per il costo computazionale.

References

[1] Sara Grassi, Hui Huang, Lorenzo Pareschi, and Jinniao Qiu. Mean-field particle swarm optimization. In *Modeling and Simulation for Collective Dynamics*, volume 40 of *IMS Lecture Notes Series*, pages 127–194. World Scientific, 2023.

[2] Giacomo Borghi, Michael Herty, Lorenzo Pareschi, *Constrained consensus-based optimization*, arXiv:2111.10571 [math.OC], 2021.

[3] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming, vol. 144, no. 1–2, pp. 1–38, 2014.

[4] Andrea Roli. *An Introduction to Metaheuristics.* Tutorial disponibile in formato PDF, Università degli Studi di Bologna, Dipartimento di Elettronica e Informatica (DEIS), 2002.

[5] Lorenzo Pareschi, Elisa Iacomini, Luca Saluzzi. *Dispense del corso "Metodi di ottimizzazione stocastici".* Università di Ferrara, anno accademico 2025/2026.