POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Dipartimento di Ingegneria Meccanica ed Aerospaziale

Tesi di Laurea Magistrale

Metodi di ottimizzazione della disposizione dei sensori per la ricostruzione del campo di spostamenti mediante iFEM

Relatori

Prof. Marco GHERLONE

Dott. Marco ESPOSITO

Candidato

Davide CASSIANO

Anno accademico 2022/2023

Sommario

Nel seguente lavoro di tesi verrà affrontata l'implementazione del Particle Swarm Optimization e del Whale Optimization Algorithm, due algoritmi di ottimizzazione metaeuristica, come possibili alternative all'ormai ben consolidato GA. Lo scopo è quello di individuare una configurazione ottimale di sensori per la misura della deformazione su di una piastra in composito irrigidita. Ciò servirà alla ricostruzione del campo di spostamenti mediante iFEM, ovvero, una metodologia di Shape Sensing che si fonda sulla minimizzazione di un funzionale, basato sull'errore ai minimi quadrati tra le deformazioni misurate sperimentalmente e quelle analitiche. Infatti, si evince che sia la collocazione dei sensori che il loro numero influenzano l'accuratezza con la quale la deformata della struttura viene ricreata.
Si andrà dunque ad introdurre il concetto di metaeuristica, ripercorrendo da un punto di vista storico, quali sono stati i passi effettuati in letteratura nel campo dell'ottimizzazione. In particolare, ci si soffermerà sugli aspetti analitici delle due metaeuristiche considerate: il PSO ed il WOA.
Dopo aver introdotto l'iFEM e relativi elementi utilizzati nel modello numerico, quali iTM2D0, iMIN2 e iQS4, si passerà alla spiegazione dei due algoritmi implementati, sia nella loro versione continua che binaria.
Infine verrà eseguita un'ottimizzazione dei sensori presenti sulla struttura, al fine di ridurne il numero garantendo al contempo un'adeguata accuratezza nella ricostruzione della deformata.

Ringraziamenti

Ringrazio il Prof. Marco Gherlone ed il Dott. Marco Esposito, che in questi mesi di lavoro hanno saputo guidarmi, con indispensabili consigli ed infinita disponibilità, nelle ricerche e nella stesura dell'elaborato.
Un ringraziamento particolare va a tutti i miei amici, a quelli di "giù" ed a quelli di "su": il destino si divertì con quel fortuito incontro sul 10, ancor prima di varcare le porte dell'università...da quel giorno, mille peripezie hanno accompagnato il nostro cammino, facendoci divertire, piangere ma soprattutto crescere.
Infine vorrei, con dei versi, ringraziare la mia famiglia perchè credo non esistano frasi adatte a descrivere quanto vi possa amare.

"Al vento di Levante,
che mi ha dato alla luce
A quello di Ponente,
che alla persistenza induce
Ed al soffio di Tramontana,
che con grazia disarmante,
ogni scorcio di Scirocco allontana"

Indice

Elenco delle tabelle VIII

Elenco delle figure IX

Acronyms XIII

1 Ottimizzazione 1
- 1.1 Introduzione all'ottimizzazione . . . 1
- 1.2 Classificazione degli algoritmi di ottimizzazione: DBAs e SBAs . . . 3
- 1.3 Euristica e metaeuristica nel campo dell'ottimizzazione . . . 4
  - 1.3.1 Pre-theoretical period . . . 5
  - 1.3.2 Early period . . . 6
  - 1.3.3 Method-centric period . . . 7
  - 1.3.4 Framework-centric period . . . 7
  - 1.3.5 Scientific period . . . 8
  - 1.3.6 Metauristica nature-inspired . . . 8
  - 1.3.7 Framework generale di una metaeuristica nature-inspired . . . 9
- 1.4 Algoritmi Genetici . . . 11
  - 1.4.1 Elementi principali di un GA . . . 11
  - 1.4.2 Operatori genetici . . . 12
  - 1.4.3 Algoritmo genetico di base . . . 13
- 1.5 Particle Swarm Optimization . . . 15
  - 1.5.1 Reti sociali . . . 16
  - 1.5.2 L'algoritmo . . . 17
  - 1.5.3 Modello del comportamento sociale . . . 20
  - 1.5.4 Guaranteed Convergence PSO . . . 24
  - 1.5.5 Binary PSO . . . 26
- 1.6 Whale Optimization Algorithm . . . 29
  - 1.6.1 Accerchiamento della preda . . . 30
  - 1.6.2 Bubble-net attack . . . 31
  - 1.6.3 Ricerca della preda . . . 32

1.6.4 Algoritmo WOA 32
1.6.5 Binary WOA 33

2 Shape sensing 35
2.1 Metodo modale 36
2.2 Teoria di Ko 37
2.3 Metodo agli elementi finiti inversi 1D 38
2.3.1 Elemento iTM2D0 41
2.4 Metodo agli elementi finiti inversi 2D 42
2.4.1 Elemento iMIN3 48
2.4.2 Elemento iQS4 50

3 Ottimizzazione della disposizione dei sensori mediante PSO e WOA 53
3.1 Spazio di ricerca 54
3.1.1 Ottimizzazione continua 54
3.1.2 Ottimizzazione discreta 55
3.2 Estensione del dominio 56
3.3 Ottimizzazione vincolata 58
3.4 Arrotondamento 59
3.5 Implementazione del PSO 59
3.5.1 Versione continua 60
3.5.2 Versione binaria 61
3.6 Implementazione del WOA 62
3.6.1 Versione continua 62
3.6.2 Versione binaria 63
3.7 Comparazione tra GA, PSO e WOA 65
3.7.1 Ottimizzazione con fibre e rosette 65
3.7.2 Ottimizzazione con sole rosette 67

4 Ricostruzione del campo di spostamenti di una piastra irrigidita in composito 70
4.1 Setup sperimentale 70
4.2 Modelli numerici 73
4.3 Configurazione sensoristica 74
4.4 Risultati sperimentali 77
4.4.1 Configurazione iniziale 77
4.4.2 Configurazioni con numero ridotto di sensori 78

5 Conclusioni e sviluppi futuri 81

A Funzioni di forma elemento iTM2D0 83

B Funzioni di forma elemento iQS4 84

C Risultati ottimizzazione per la disposizione di fibre e rosette 86

D Risultati ottimizzazione per la disposizione delle rosette 103

Bibliografia 112

Elenco delle tabelle

1.1 Comparazione tra terminologia naturale e di un GA 12
1.2 Funzioni di trasferimento 28

3.1 Esempio di ID per rosette e fibre 55
3.2 Matrice di estensione: limite superiore 57
3.3 Matrice di estensione: limite inferiore 57
3.4 Primo setup per l'ottimizzazione della disposizione di fibre e rosette 66
3.5 Secondo setup per l'ottimizzazione della disposizione di fibre e rosette 66
3.6 Medie della fitness e delle valutazioni iFEM: 14 individui 66
3.7 Medie della fitness e delle valutazioni iFEM: 28 individui 67
3.8 Setup per l'ottimizzazione delle rosette 68
3.9 Medie della fitness e delle valutazioni iFEM per la disposizione delle rosette 68

4.1 Proprietà meccaniche TWILL T-300 70
4.2 Spostamenti nei punti individuati 77
4.3 Media degli errori relativi: FEM e iFEM 78
4.4 ERMS numerico dello spostamento trasversale delle tre configurazioni considerate 78
4.5 Spostamenti nei punti individuati 80
4.6 Media degli errori relativi: FEM e iFEM 80

C.1 Risultati PSO: 14 individui e 500 iterazioni 94
C.2 Risultati WOA: 14 individui e 500 iterazioni 94
C.3 Risultati GA: 100 individui e 33 iterazioni 95
C.4 Risultati PSO 28: individui e 250 iterazioni 98
C.5 Risultati WOA 28: individui e 250 iterazioni 99
C.6 Risultati GA: 50 individui e 68 iterazioni 99

D.1 Run PSO 107
D.2 Run WOA 107
D.3 Run GA 108

Elenco delle figure

1.1 Esempio di minimo locale e di minimo globale 2
1.2 Classificazione degli algoritmi deterministici [7] 3
1.3 Classificazione degli algoritmi metaeuristici [2] 9
1.4 Flow-chart di un algoritmo nature-inspired 10
1.5 Esempio di uno stormo di uccelli 16
1.6 Tipiche reti sociali [22] 17
1.7 Illustrazione geometrica dell'aggiornamento delle velocità e delle posizioni per una particella bi-dimensionale [22] 21
1.8 Funzione sigmoidea 27
1.9 Funzioni di trasferimento: s-shape e v-shape 29
1.10 Bubble-net feeding behavior [28] 30
1.11 Approcci del bubble-net attack [28] 32

2.1 Geometria e posizione dei sensori 37
2.2 Notazione per la trave 38
2.3 Elemento iTM2d0 41
2.4 Notazione per la piastra 42
2.5 Rappresentazione grafica delle deformazioni misurate 45
2.6 Elemento iMIN3 49
2.7 Elemento iQS4 51

3.1 Interfaccia tra funzione obiettivo e framework di ottimizzazione . . 54
3.2 Rappresentazione grafica di una possibile configurazione di sensori . 55
3.3 Esempio di codifica a 9 bit con individui costituiti da sei sensori . . 56
3.4 Spiegazione delle tre metodologie di handling boundary constraints [36] 56
3.5 Esempio di rimappatura 57
3.6 Individuo del PSO 60
3.7 Individuo del WOA 62
3.8 Funzione di trasferimento: U-shape 65

3.9 Comparazione ottimizzazione di fibre e rosette: 14 individui e 500 iterazioni . . . 67
3.10 Comparazione ottimizzazione rosette . . . 68

4.1 Geometria in pianta del pannello . . . 71
4.2 Sezione trasversale corrente . . . 71
4.3 Direzioni fibre . . . 71
4.4 Semplice appoggio realizzato in laboratorio . . . 72
4.5 Sistema di carico . . . 72
4.6 Modello FEM . . . 73
4.7 Disposizione dei sensori . . . 74
4.8 LUNA® high-definition distributed fiber optic strain sensing [39] . . . 74
4.9 Schema del funzionamento di un OFDR [41] . . . 75
4.10 Disposizione delle fibre lungo l'anima dell'irrigidimento . . . 75
4.11 Dettaglio dei sensori utilizzati . . . 76
4.12 Posizione degli LVDT e del punto di applicazione del carico . . . 76
4.13 Campo di spostamenti: componente trasversale w . . . 77
4.14 Configurazione ottimizzata: 5 fibre e 5 rosette . . . 79
4.15 Configurazione ottimizzata: 5 fibre e 3 rosette . . . 79
4.16 Configurazione ottimizzata: 3 fibre e 3 rosette . . . 79

C.1 Run 1: 50 individui GA e 28 individui PSO/WOA . . . 86
C.2 Run 2: 50 individui GA e 28 individui PSO/WOA . . . 87
C.3 Run 3: 50 individui GA e 28 individui PSO/WOA . . . 87
C.4 Run 4: 50 individui GA e 28 individui PSO/WOA . . . 87
C.5 Run 5: 50 individui GA e 28 individui PSO/WOA . . . 88
C.6 Run 6: 50 individui GA e 28 individui PSO/WOA . . . 88
C.7 Run 7: 50 individui GA e 28 individui PSO/WOA . . . 88
C.8 Run 8: 50 individui GA e 28 individui PSO/WOA . . . 89
C.9 Run 9: 50 individui GA e 28 individui PSO/WOA . . . 89
C.10 Run 10: 50 individui GA e 28 individui PSO/WOA . . . 89
C.11 Run 1: 100 individui GA e 14 individui PSO/WOA . . . 90
C.12 Run 2: 100 individui GA e 14 individui PSO/WOA . . . 90
C.13 Run 3: 100 individui GA e 14 individui PSO/WOA . . . 90
C.14 Run 4: 100 individui GA e 14 individui PSO/WOA . . . 91
C.15 Run 5: 100 individui GA e 14 individui PSO/WOA . . . 91
C.16 Run 6: 100 individui GA e 14 individui PSO/WOA . . . 91
C.17 Run 7: 100 individui GA e 14 individui PSO/WOA . . . 92
C.18 Run 8: 100 individui GA e 14 individui PSO/WOA . . . 92
C.19 Run 9: 100 individui GA e 14 individui PSO/WOA . . . 92
C.20 Run 10: 100 individui GA e 14 individui PSO/WOA . . . 93

C.21 Run 1 con 14 individui: PSO e WOA 95
C.22 Run 2 con 14 individui: PSO e WOA 95
C.23 Run 3 con 14 individui: PSO e WOA 96
C.24 Run 4 con 14 individui: PSO e WOA 96
C.25 Run 5 con 14 individui: PSO e WOA 96
C.26 Run 6 con 14 individui: PSO e WOA 97
C.27 Run 7 con 14 individui: PSO e WOA 97
C.28 Run 8 con 14 individui: PSO e WOA 97
C.29 Run 9 con 14 individui: PSO e WOA 98
C.30 Run 10 con 14 individui: PSO e WOA 98
C.31 Run 1 con 28 individui: PSO e WOA 99
C.32 Run 2 con 28 individui: PSO e WOA 100
C.33 Run 3 con 28 individui: PSO e WOA 100
C.34 Run 4 con 28 individui: PSO e WOA 100
C.35 Run 5 con 28 individui: PSO e WOA 101
C.36 Run 6 con 28 individui: PSO e WOA 101
C.37 Run 7 con 28 individui: PSO e WOA 101
C.38 Run 8 con 28 individui: PSO e WOA 102
C.39 Run 9 con 28 individui: PSO e WOA 102
C.40 Run 10 con 28 individui: PSO e WOA 102

D.1 Run 1: 200 individui GA e 200 individui PSO/WOA 103
D.2 Run 2: 200 individui GA e 200 individui PSO/WOA 104
D.3 Run 3: 200 individui GA e 200 individui PSO/WOA 104
D.4 Run 4: 200 individui GA e 200 individui PSO/WOA 104
D.5 Run 5: 200 individui GA e 200 individui PSO/WOA 105
D.6 Run 6: 200 individui GA e 200 individui PSO/WOA 105
D.7 Run 7: 200 individui GA e 200 individui PSO/WOA 105
D.8 Run 8: 200 individui GA e 200 individui PSO/WOA 106
D.9 Run 9: 200 individui GA e 200 individui PSO/WOA 106
D.10 Run 10: 200 individui GA e 200 individui PSO/WOA 106
D.11 Run 1 con 200 individui: PSO e WOA 108
D.12 Run 2 con 200 individui: PSO e WOA 108
D.13 Run 3 con 200 individui: PSO e WOA 109
D.14 Run 4 con 200 individui: PSO e WOA 109
D.15 Run 5 con 200 individui: PSO e WOA 109
D.16 Run 6 con 200 individui: PSO e WOA 110
D.17 Run 7 con 200 individui: PSO e WOA 110
D.18 Run 8 con 200 individui: PSO e WOA 110
D.19 Run 9 con 200 individui: PSO e WOA 111
D.20 Run 10 con 200 individui: PSO e WOA 111

Acronyms

iFEM
Inverse Finite Element Method

FEM
Finite Element Method

DBAs
Deterministic-Based Algorithms

SBAs
Stocastic-Based Algorithms

OR
Operations Research

GA
Genetic Algorithm

PSO
Particle Swarm Optimization

WOA
Whale Optimization Algorithm

ES
Evolution Strategy

DE
Differential Evolution

GP

Genetic Programming

SI

Swarm Intelligence

ACO

Ant Colony Optimization

ABC

Artificial Bee Colony

SA

Simulated Annealing

GSA

Gravitational Search Algorithm

QGA

Quantum Inspired Genetic Algorithm

HS

Harmony Search

FA

Firework Algorithm

ICA

Imperial Competitive Algorithm

GCPSO

Guaranteed Converged PSO

BPSO

Binary PSO

BWOA

Binary WOA

SHM

Structural Health Monitoring

FSDT

First Shear Deformation Theory

SEA

Smoothing Element Analysis

DOF

Degree Of Freedom

EBBT

Euler–Bernoulli beam theory

FBG

Fiber Bragg Grating

BE-WOA

Binary Enhanced Whale Optimization Alorithm

OFDR

Optical Frequency Domain Reflectometry

Capitolo 1

Ottimizzazione

Uno dei principi fondamentali della realtà a noi conosciuta è la ricerca di uno stato ottimale: in economia, i profitti e le vendite devono essere massimizzati ed i costi ridotti il più possibile, una molecola assume nello spazio una configurazione in cui lo stato energetico risulti il più basso possibile, nella teoria Darwiniana una specie ben adattata prevale nella lotta alla sopravvivenza.
L'ottimizzazione può dunque essere descritta come il processo di selezione di alcuni elementi al fine di ottenere i migliori risultati possibili [1].
Vari sono i campi di applicazione come la robotica, le reti informatiche, la sicurezza, la progettazione ingegneristica, la gestione dei dati, il data mining ed indipendentemente dalla natura del problema rappresenta un tema di profondo interesse nella attività di ricerca [2].

1.1 Introduzione all'ottimizzazione

L'ottimizzazione matematica è una branca della matematica applicata e delle scienze informatiche che si occupa della selezione della soluzione ottimale per una particolare funzione matematica f_0, nota come *funzione obiettivo*

$$f_0 : \mathcal{R}^n \to \mathcal{R}$$

con lo scopo di minimizzarne o massimizzarne il risultato. La funzione obiettivo è tipicamente soggetta a dei *vincoli* espressi attraverso la funzione f_i

$$f_i : \mathcal{R}^n \to \mathcal{R}$$

applicati alle *variabili di ottimizzazione* del problema $\mathbf{x}$

$$\mathbf{x} = (x_1, \ldots, x_n) \quad \text{dove} \quad \mathbf{x} \in \mathcal{R}^n$$

In generale, un problema di ottimizzazione matematica presenta dunque la seguente forma [3]

$$\begin{aligned} &\min_{x \in \mathcal{R}^n} && f_0(x) \\ &\text{soggetto a} && f_i(x) \leq b_i,\ i = 1, \ldots, m. \end{aligned}$$

dove le costanti $b_1, \ldots, b_m$ rappresentano i *limiti* dei vincoli.
Un vettore $\mathbf{x}^*$ è definito *ottimo* o *soluzione* del problema, se permette di ottenere la più piccola valutazione della funzione obiettivo fra tutti i vettori appartenenti al dominio di f_0, soddisfando al contempo i limiti imposti dai vincoli.
In particolare [4], definendo un parametro $\epsilon > 0$ sufficientemente piccolo è possibile definire un *minimo locale* se

$$f_0(x^*) \leq f_0(x), \qquad \forall \mathbf{x} \in \mathcal{R}^n \backslash \{\mathbf{x}^*\} \quad \text{con} \quad ||\mathbf{x} - \mathbf{x}^*|| < \epsilon$$

oppure un *minimo globale* se

$$f_0(x^*) \leq f_0(x), \qquad \forall \mathbf{x} \in \mathcal{R}^n \backslash \{\mathbf{x}^*\}$$

Vengono in Figura 1.1 illustrate le definizioni di minimo locale e globale.

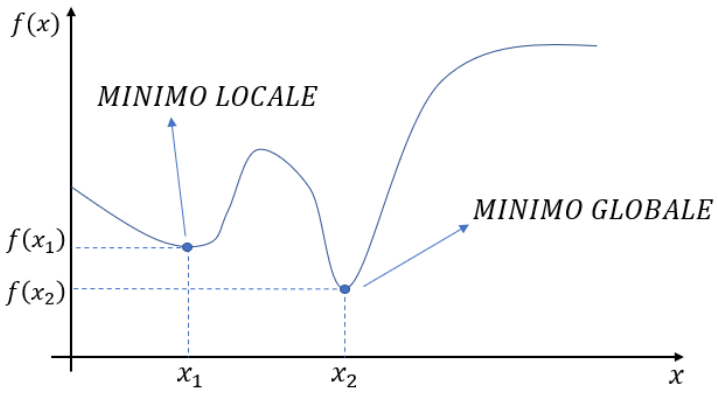

Figura 1.1: Esempio di minimo locale e di minimo globale

Un'ottimizzazione può inoltre essere *continua* o *discreta* [5]. Nei problemi continui le variabili sono definite all'interno di un dominio anch'esso continuo assumendo dunque un valore reale. Di contro, vengono definiti i problemi discreti in cui le variabili possono assumere esclusivamente valori interi in un range prestabilito come ad esempio nel calcolo combinatorio.

1.2 Classificazione degli algoritmi di ottimizzazione: DBAs e SBAs

Tradizionalmente, le tecniche di ottimizzazione vengono classificate come *stocastiche* (SBAs) o *deterministiche* (DBAs) [1].
In generale, gli algoritmi deterministici non sfruttano sequenze numeriche randomiche al fine di decidere cosa fare o come modificare i dati del problema e forniscono inoltre garanzie teoriche sul raggiungimento del minimo globale, all'interno di una certa tolleranza stabilita.
Affinché un algoritmo possa esser classificato come tale, deve risultare *completo* e *rigoroso* [6]. In particolare, un metodo è definito completo quando raggiunge un minimo globale con certezza, assumendo calcoli esatti e tempi di esecuzione indefinitamente lunghi. Per cui è possibile ipotizzare con assoluta certezza che dopo un certo periodo di tempo è stato individuato un minimo globale in maniera approssimata, entro le tolleranze definite.
Un metodo è rigoroso quando invece converge all'ottimo locale, sempre con certezza, entro un periodo di tempo finito ed entro le tolleranze prestabilite anche in presenza di errori di arrotondamento.
Caratteristica importante di questa famiglia di algoritmi è l'utilizzo, durante la fase di ricerca, di informazioni riguardo la funzione obiettivo, come ad esempio le derivate. In Figura 1.2 è riportata una classificazione di alcuni algoritmi deterministici.

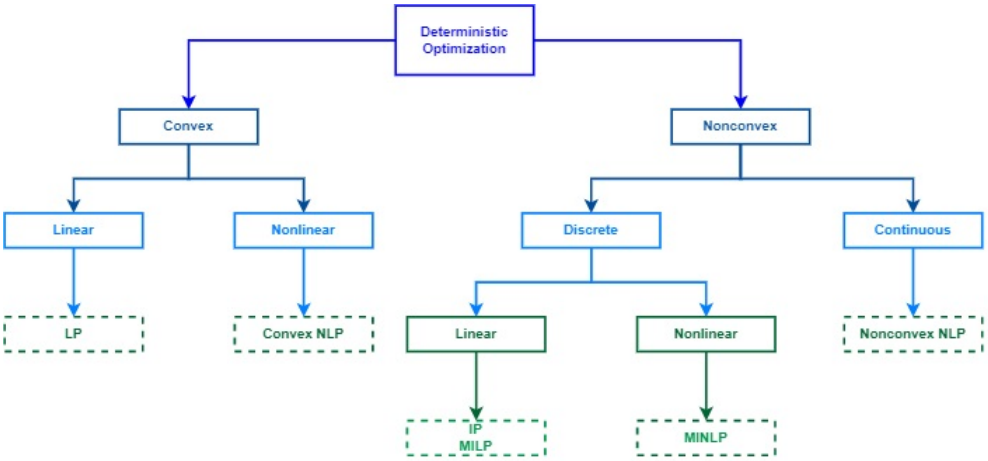

Figura 1.2: Classificazione degli algoritmi deterministici [7]

Invece, l'approccio al problema di tipo probabilistico genera ed usa variabili di tipo casuale risultando dunque meno dipendente dalla formulazione del problema.
Per la propria natura, gli SBAs riescono ad esplorare il dominio della funzione

obiettivo in modo tale da evitare i punti di minimo locale in maniera più efficiente [8] rispetto alla controparte deterministica. Di contro, presentano una convergenza più lenta e non garantiscono il raggiungimento di un minimo globale [9], ponendo dunque una forte limitazione in quelle applicazioni in cui il tempo risulta essere un fattore importante. Principale esempio di tali tecniche di ottimizzazione è il metodo Monte Carlo che antepone alla correttezza del risultato finale il costo computazionale.
Queste problematiche, sommate al continuo cambiamento della natura dei problemi di ottimizzazione hanno portato allo sviluppo di due nuove scuole di pensiero da introdurre nel mondo della stocastica: l'*euristica* e la *metaeuristica*.

1.3 Euristica e metaeuristica nel campo dell'ottimizzazione

Storicamente, è possibile individuare 5 periodi nei quali i concetti euristici e metaeuristici sono stati sviluppati [10]:

1. *Pre-theoretical*: fino all'inizi degli anni '40 l'euristica e la metaeuristica vennero utilizzate ma non studiate in maniera formale.

2. *Early period*: tra il 1940 ed il 1980 l'euristica incominciò ad essere studiata in maniera formale.

3. *Method-centric period*: tra il 1980 e l'inizio del nuovo millennio la metaeuristica prese il sopravvento con l'introduzione di molte tipologie di algoritmi differenti.

4. *Framework-centric period*: dal 2000 fino ai giorni nostri la metaeuristica non viene più concepita come metodo ma come vera e proprio struttura da dare all'algoritmo.

5. *Scientific period*: nel futuro la metaeuristica diventerà una vera e propria scienza anziché un arte.

Sörensen e Glover [10] definiscono la metaeuristica nel seguente modo

"Una metaeuristica è un *algorithmic framework*[1] di alto livello indipendente dal problema, che fornisce un insieme di linee guida o strategie per sviluppare algoritmi di ottimizzazione euristica. Il termine viene utilizzato anche per indicare

[1]Nella programmazione, un algorithmic framework è una struttura aggiuntiva del codice scritta dall'utente che modifica in maniera selettiva l'algoritmo originale, fornendone così una specifica aggiuntiva

un'implementazione specifica di un algoritmo di ottimizzazione euristica in base alle linee guida espresse in tale framework"

Come è possibile osservare, vengono date due definizioni diverse al termine metaeuristica. Una è quella inerente al concepire la metaeuristica come un framework di alto livello svincolato dalla tipologia del problema, ovvero, come un insieme di concetti e strategie applicabili in qualsiasi campo di ottimizzazione: in questo senso, la ricerca dell'ottimo viene affrontata attraverso l'utilizzo di diversi *operatori* o *individui* assieme ad un *operatore di perturbazione* che aiuta a perturbare e quindi ad allontanare gli operatori stessi da quello che potrebbe essere un ottimo locale.
La seconda definizione si riferisce alla metaeuristica come implementazione specifica di un algortimo, basato su una ben definita struttura (o su differenti concetti provenienti da differenti ambiti) che ha il compito di individuare la soluzione di uno specifico problema di ottimizzazione. Nei seguenti sottoparagrafi verrà utilizzato il termine *framework metaeuristico* per far riferimento alla prima definizione di metaeuristica e *algoritmo metaeuristico* per la sua seconda declinazione.

1.3.1 Pre-theoretical period

Fin dall'adolescenza l'uomo risolve differenti tipologie di problemi, molti dei quali richiedono un continuo perfezionamento della soluzione. La capacità di risolvere un problema di ottimizzazione nella maniera più veloce ed adeguata possibile è un fattore determinante nella probabilità di sopravvivenza delle varie specie e nel corso degli anni è andata via via ad evolversi, seppur facendo sopravvivere un'attitudine di natura *euristica* e non ottimale. Infatti, per euristica si intende un insieme di strategie, tecniche o procedimenti inventivi che si affidano all'intuito e allo stato temporaneo delle circostanze al fine di generare nuova conoscenza [11].
Per cui, l'obiettivo di un algoritmo euristico è quello di produrre una soluzione sufficientemente adeguata ad un problema specifico sfruttandone le informazioni relative in un tempo ragionevole. Chiaramente, tale soluzione può non essere la migliore ma è comunque valida in quanto non richiede un tempo proibitivo per trovarla [12].
Data la diversa natura dei problemi, la mente umana ha inoltre la capacità di usare strategie *metaeuristiche*. Ad esempio, quando si è davanti ad un nuovo problema da risolvere del quale la soluzione non è subito identificabile, cerca automaticamente di individuare problemi simili già risolti nel passato tentando di ricavarne delle regole da applicare per la nuova soluzione. Questa strategia è nota come *Learning by Analogy*, utilizzata molto nel *Machine Learning*.

1.3.2 Early period

Il matematico ungherese György Pólya pubblicò nel 1945 il libro *"How to solve it"*, nel quale propose diverse strategie non strettamente inerenti al campo dell'ottimizzazione ma comunque applicabili come il *principio dell'analogia*, il *principio dell'induzione* ed il *principio del problema ausiliario*. Ciò diede inizio allo sviluppo della *ricerca operativa* e delle prime tecniche di ottimizzazione con una certa formalità nella loro formulazione. Basti pensare che alcuni dei principi di Pólya, anche se non in maniera diretta, vengono tutt'ora usati.
In questo periodo nacquero inoltre alcune idee inerenti ad algoritmi di alto livello come ad esempio gli *algoritmi costruttivi* nei quali si parte da una soluzione vuota ed attraverso una sequenza di soluzioni parziali da aggiungere iterativamente si determina una soluzione completa. Esistono diverse strategie di selezione:

- *Selezione greedy*: ad ogni iterazione viene selezionata la miglior valutazione ed aggiunta a quelle precedenti. Ad esempio, è possibile con questa filosofia di selezione determinare il minor numero di monete per ottenere un certo valore in denaro.

- *Selezione Regret*: ad ogni iterazione viene selezionata la soluzione che se non fosse stata scelta avrebbe portato ad un costo di penalizzazione elevato. Ad esempio, in ambito aziendale è un'euristica che consente al lavoratore di prendere una decisione proiettandosi nel futuro, in età avanzata, visualizzando se il rimpianto di aver perso un'opportunità lo perseguiterebbe, rispetto a quello di aver colto l'opportunità ed aver fallito.

Verso la fine degli anni '50 iniziarono a nascere inoltre una serie di metodi che al giorno d'oggi costituiscono la base degli *algoritmi evolutivi* e che servirono più che altro a studiare il fenomeno dell'evoluzione.
L'idea che tutto ciò potesse essere usato come tecnica di ottimizzazione nacque agli inizi degli anni 60' quando vari ricercatori incominciarono a sviluppare algoritmi per la risoluzione di tali problemi. Uno dei primi metodi fu *la strategia dell'evoluzione* dove una soluzione, definita genitore, viene mutata e la migliore fra le due diventa il genitore della prossima mutazione.
L'inizio dell'era degli algoritmi evolutivi si ebbe poi nel 75' quando venne introdotto il cosiddetto *Teorema degli schemi*, riportato nel Paragrafo 1.4.3, di John Holland secondo cui sotto determinate ipotesi, gli individui con alti valori di fitness tendono a crescere esponenzialmente nella popolazione attraverso il meccanismo dell'incrocio, assicurando così la convergenza dell'algoritmo genetico verso una soluzione ottimale.

1.3.3 Method-centric period

A partire dagli anni '80, la programmazione evolutiva incominciò a diventare molto popolare, anche grazie alla pubblicazione del libro *"Genetic Algorithms in Search"* di Andrew Goldberg, con conseguente crescita esponenziale della letteratura inerente a tale campo.
Inoltre in questo periodo venne coniato per la prima volta il termine metaeuristico anche se la consapevolezza che le metaeuristiche potevano e dovevano essere viste come strutture generali piuttosto che come algoritmi specifici, ovvero un insieme di operazioni da seguire passo dopo passo, sarebbe arrivata durante il periodo successivo, ovvero il framework-centric period.
In questi anni, venne introdotto un framework per il problem-solving non basato sull'evoluzione naturale: il *Simulated Annealing*. Il concetto di annealing ("ricottura") deriva dalla scienza dei metalli, dov'è usato per descrivere il processo di eliminazione di difetti reticolari dai cristalli tramite riscaldamento seguito da lento raffreddamento.
Altro framework fu il *Tabu Search*: questo algoritmo di ricerca locale usa un elenco che rappresenta un set di potenziali soluzioni, tipicamente le ultime mosse eseguite nel cammino di ricerca, da non visitare per una serie di passaggi in quanto si rischierebbe di ricadere nuovamente nel medesimo minimo locale.
Ancora più interessanti furono le *reti neurali* che riflettono il funzionamento del cervello umano. Tale modello è costituito da un gruppo di interconnessioni di informazioni costituite da neuroni artificiali e processi. Nella maggior parte dei casi una rete neurale artificiale è un sistema adattivo che cambia la propria struttura in base a informazioni esterne o interne che scorrono attraverso la rete stessa durante la fase di apprendimento.

1.3.4 Framework-centric period

L'intuizione che la metaeuristica potesse essere descritta più utilmente come struttura algoritmica di alto livello, piuttosto che come semplice algoritmo, fu una cosa del tutto naturale. Il principale indicatore dell'avvento di questo nuovo paradigma fu la crescente popolarità delle cosiddette metaeuristiche "ibride" dei primi anni 2000 da non considerare come mera combinazione di quest'ultime. La sua concezione come struttura di alto livello ha permesso infatti ai ricercatori di combinare una metaeuristica con qualsiasi metodo ausiliario disponibile come ad esempio la programmazione lineare. La combinazione di metaeuristiche e metodi deterministici venne poi definita *"mateuristica"*.
Per cui il concetto di framework metaeuristico implicava che le metaeuristiche non fossero altro che un insieme più o meno coerente di idee che, ovviamente, potevano essere liberamente combinate fra loro.

1.3.5 Scientific period

Anche se la comunità scientifica ha compiuto notevoli progressi nella ricerca al fine di comprendere il comportamento fondamentale della metaeuristica, il principale obiettivo rimane sempre quello di aumentare le prestazioni. La ricerca è considerata buona se e solo se produce un algoritmo che "performa" meglio rispetto un altro. Tuttavia, quello che si osserva è una lenta trasformazione di quello che si considera a tutti gli effetti una sorta di gioco in una vera e propria scienza imponendo una maggior attenzione alla comprensione piuttosto che alle prestazioni dell'algoritmo.

1.3.6 Metauristica nature-inspired

La natura si dimostra essere un perfetto esempio di problem solving adattivo [1] a tal punto che molte sono state le strutture di alto livello ideate nella metaeuristica, prendendo come esempio il modo in cui avvengono determinati fenomeni naturali o biologici.
La metaeuristica nature-inspired può essere classificata in quattro categorie principali: evolution-based, swarm-based, physics-based e human-based. I metodi evolution-based si ispirano alle leggi dell'evoluzione naturale basate sulla teoria fondamentale dell'evoluzione darwiniana. Rappresentano dunque una famiglia degli algoritmi biologically-inspired [13] che si basano sulla gestione delle popolazioni e sui processi di replicazione, mutazione ed infine selezione. Tra questi algoritmi troviamo gli algoritmi genetici (GA), gli Evolution Strategy (ES), la Differential Evolution (DE) ed infine la Genetic Programming (GP).
I metodi swarm-based [14] si basano sulla cosiddetta Swarm Intelligence (SI) definita come

"Intelligenza collettiva che emerge da un gruppo di singoli individui"

Sono dunque metodologie che simulano il comportamento sociale e collettivo di un gruppo di animali come ad esempio uccelli, insetti e pesci.
Due aspetti principali considerati come proprietà fondamentali della SI sono l'auto-organizzazione e la divisione dei compiti. Per auto-organizzazione si intende la capacità del sistema di far evolvere i suoi agenti senza la presenza di un aiuto esterno. Invece, la seconda proprietà è definita come il simultaneo svolgimento dei vari tasks da parte degli individui.
Alcuni algoritmi che rientrano in questa categoria sono il Particle Swarm Optimization (PSO), l'Ant Colony Optimization (ACO), l'Artifical Bee Colony (ABC) ed il Whale Optimization Algorithm (WOA).
Gli algoritmi physics-based sono invece sviluppati con l'idea di simulare le leggi della fisica osservate nell'universo. Tra i più popolari troviamo il Simulated Annealing (SA), il Gravitational Search Algorithm (GSA) ed il Quantum-Inspired Genetic

Algorithm (QGA).
Infine, gli algoritmi human-based prendono spunto dal comportamento dell'uomo, dal suo stile di vita e dalla sua percezione. Fra i metodi più conosciuti in letteratura troviamo l'Harmony Search (HS), il Firework Algoritm (FA) e l'Imperialist Competitive Algorithm (ICA).

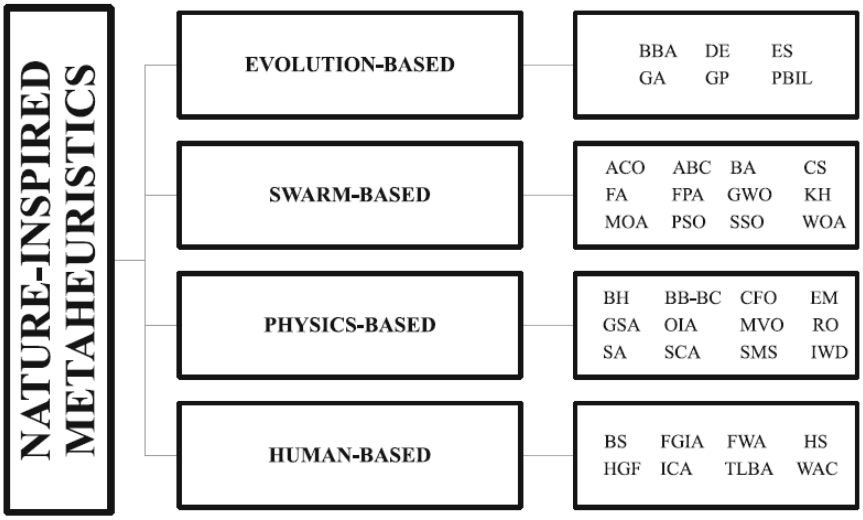

Figura 1.3: Classificazione degli algoritmi metaeuristici [2]

1.3.7 Framework generale di una metaeuristica nature-inspired

La maggior parte delle metaeuristiche nature-inspired sono modellate come algoritmi population-based il che implica che il framework generale impiegato rimane pressoché identico indipendetemente dalla natura del fenomeno dalla quale l'algoritmo stesso è ispirato. In Figure 2.5 è possibile osservare il flow-chart di un generico nature-inspired framework.
Tipicamente, il primo step prevede la generazione casuale di una *popolazione* costituita da **n** ipotetiche soluzioni, note come *individui*

$$\mathbf{X_i} = \{x_{i,1}, \ldots, x_{i,n}\}$$

dove n viene definita come *dimensionality* dello spazio delle soluzioni, ovvero il numero di individui di ogni popolazione. Tale valore può essere fisso o variabile [13]. A ciascuna di queste soluzioni viene poi assegnato il corrispondente *valore di qualità* o *fitness* inerente alla valutazione della funzione obiettivo

$$f_{i,j} = f(x_{i,j})$$

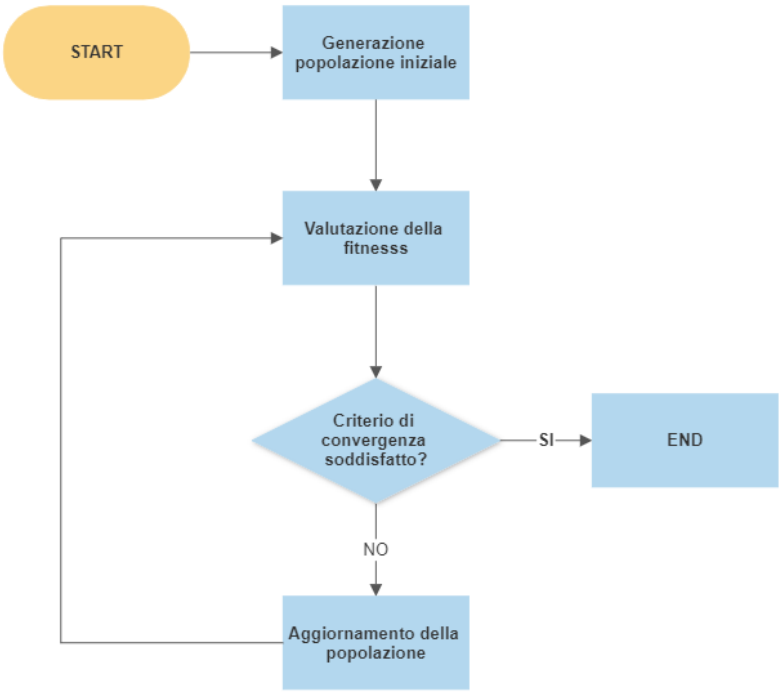

Figura 1.4: Flow-chart di un algoritmo nature-inspired

con j=1, . . . , n e i=1, . . . , N dove N sta ad indicare il numero di popolazioni generate, ovvero delle iterazioni.
Ciascun metodo nature-inspired segue poi uno schema iterativo di ricerca attraverso il quale le nuove soluzioni o *candidati* sono generate modificando l'individui già presenti ed appartenenti alla generazione precedente. Tale schema è descritto dalla seguente equazione

$$\mathbf{X_{i+1}} = \mathbf{X_i} + \Delta\mathbf{X_i} \tag{1.1}$$

dove $\mathbf{X_{i+1}}$ rappresenta la nuova possibile soluzione e $\Delta\mathbf{X_i}$ il vettore inerente all'aggiornamento della variabile diverso a seconda della metaeuritica considerata.
L'intero processo è ripetuto iterativamente finché non viene soddisfatto un particolare criterio, tipicamente un numero di iterazioni massimo. Si ottiene a questo punto la miglior soluzione che chiaramente rappresenta un'approssimazione dell'ottimo globale.

1.4 Algoritmi Genetici

L'informazione ereditaria è trasmessa sotto forma di unità distinte note come *geni* [15] costituiti da una sequenza di DNA. Un gene, quindi, è formato da una sequenza di nucleotidi del DNA che custodisce tutte le informazioni utili a produrre una specifica molecola. L'insieme di tutti i geni presenti in un intero organismo prende poi il nome di *genoma*.
Quest'ultimi sono poi contenuti all'interno di piccole unità di DNA separate chiamate cromosomi e situati all'interno del nucleo delle cellule eucariote tipicamente a forma di bastoncino e solo presenti durante la fase di divisione cellulare. A causa delle mutazioni avvenute durante il processo evolutivo di una specie, un gene può presentarsi con delle varianti, dette *alleli*. Negli organismi diploidi, condizione in cui nelle cellule somatiche di un organismo vivente sono presenti solo due copie per ogni cromosoma, definite cromosomi omologhi, ogni gene proviene sia dall'omologo materno che dall'omologo paterno. Ciò significa che in un individuo diploide, il *carattere*, ovvero il fattore ereditario legato ad una particolare caratteristica morfologica o fisiologia dell'individuo, si può presentare sempre con due alleli che occupano la medesima posizione o *locus* nei due cromosomi omologhi. In particolare, una specifica combinazione di alleli che controlla un carattere si definisce *genotipo* che genera una manifestazione osservabile nota come *fenotipo*.
Gli algoritmi genetici sono algoritmi basati sulla selezione naturale e sulla genetica [16]. Furono sviluppati da John Holland nel 1975 e dal suo team di ricerca all'università di Michigan con gli obiettivi di spiegare rigorosamente i processi dei sistemi naturali e di progettare sistemi artificiali capaci di replicarli.
Ecco che i sistemi biologici ed in particolare l'evoluzione sono visti come una metodologia di design innovativa rappresentando dunque una fonte di ispirazione per la risoluzione di molti problemi che richiedono, nell'implementazione di una determinata soluzione, alcune caratteristiche fondamentali come ad esempio la capacità di adattamento.

1.4.1 Elementi principali di un GA

Ogni individuo della popolazione, che rappresenta una possibile soluzione nello spazio di ricerca, viene definito *cromosoma*. Ciascun cromosoma è codificato da una stringa costituita da un numero finito e costante di *geni* generalmente binari e formati da un singolo bit o da un gruppo di bit adiacenti [17]: per cui se si hanno a disposizione N geni è possibile realizzare 2^N stringhe diverse. Ciascun gene è poi collocato in una particolare posizione definita *locus* ed ognuno di essi ha due possibili *alleli*: 0 e 1, nel caso in cui la codifica fosse di tipo binario.
Ciò che viene elaborato durante una generica iterazione, nel caso più semplice

una stringa di bit, viene definito *struttura*. Infine, una volta ottenuta la soluzione, quest'ultima viene decodificata.

Terminologia naturale	Terminologia artificiale
Cromosoma	Stringa
Gene	Bit
Allele	Valore del bit
Locus	Posizione nella stringa
Genotipo	Struttura
Fenotipo	Soluzione decodificata

Tabella 1.1: Comparazione tra terminologia naturale e di un GA

In Tabella 1.1 è riportata una comparazione tra terminologia naturale e quella inerente ad un algoritmo genetico.
Il GA processa i cromosomi generando dunque una nuova popolazione ed assegnando ad ognuno di essi un valore di fitness mediante la valutazione della funzione obiettivo.

1.4.2 Operatori genetici

Ogni popolazione è costituita da un certo numero di individui ma soltanto alcuni di essi vengono selezionati per la fase di *riproduzione* premiando tendenzialmente quelli dotati di maggiore fitness. Questi individui, noti come *genitori*, vanno a costituire il cosiddetto *mating pool* ed a seguito di opportune operazioni genetiche vengono prodotte nuove varietà (individui): tra le più importanti troviamo il cross-over, la mutazione e l'inversione.
Il *cross-over* è eseguito a partire da 2 stringhe appartenenti al mating pool

$$< A_1, A_2, \ldots, A_n >$$

$$< B_1, B_2, \ldots, B_n >$$

Dopo la scelta casuale di un locus $k \in (1, n)$, o punto di cross-over, vengono prodotte le due stringhe figlie

$$< A_1, A_2, A_k, B_{k+1}, B_{k+2}, \ldots, B_n >$$

$$< B_1, B_2, B_k, A_{k+1}, A_{k+2}, \ldots, A_n >$$

Esistono altre varianti come il cross-over ad n-punti e lo shuffle cross-over[18]. Nel primo caso, vengono scelti due punti di crossover k ed l con k≤l nelle due stringhe genitrici generando le seguenti stringhe figlie

$$< A_1, A_2, \ldots, A_k, B_{k+1}, \ldots, B_{l-1} A_l, \ldots, A_n >$$

$$< B_1, B_2, \ldots, B_k, A_{k+1}, \ldots, A_{l-1}B_l, \ldots, B_n >$$

Nel secondo caso, vengono prima mescolati i geni in maniera randomica ma identica in entrambi i genitori. Viene quindi applicato un crossover ad un punto, generando le due stringhe figlie che vengono rimescolate in maniera inversa rispetto ai genitori. Nella *mutazione* un singolo bit di una stringa viene cambiato nel valore opposto, con una probabilità prefissata. Ad esempio, la stringa

$$< A_1, A_2, A_k = 0, A_{k+1}, \ldots, A_n >$$

diventa

$$< A_1, A_2, A_k = 1, A_{k+1}, \ldots, A_n >$$

.
Nell'inversione viene scelto casualmente un punto k di inversione

$$< A_1, A_2, \ldots, A_k, A_{k+1}, \ldots, A_n >$$

ed invertita la parte finale della stringa nel seguente modo

$$< A_1, A_2, \ldots, A_k, A_n, A_{n-1}, \ldots, A_{k+1} >$$

Anche per l'inversione, come per il cross-over, esistono varianti a due o più punti di inversione.
Mentre l'incrocio assicura assieme alla replicazione il mantenimento di buoni individui per migliorare il valore della fitness, la mutazione e l'inversione permettono di mantenere un certo tasso di eterogeneità all'interno della popolazione, garantendo un ampliamento dell'esplorazione. In particolare, tutti e tre gli operatori dipendono dal caso ovvero dalla probabilità che si affida all'avvenimento del crossover o della riproduzione tipicamente maggiore di quella assegnata alla mutazione.

1.4.3 Algoritmo genetico di base

Due premesse devono essere fatte prima di procedere con l'implementazione di un GA. Anzitutto, deve essere modellato il problema andando a definire lo spazio di ricerca, la funzione obiettivo e la tipologia di codifica per poi passare alla definizione di una serie di parametri come ad esempio la dimensionalità di una popolazione e quanti individui sottoporre all'incrocio, alla mutazione o alla semplice riproduzione [17].
Prima di tutto si procede con l'inizializzazione casuale di una popolazione costituita da N cromosomi costituiti da stringhe di n bit. Si procede con il calcolo della fitness, attraverso la valutazione della funzione obiettivo f per ogni individuo.
Si ripetono le seguenti istruzioni finché una popolazione di N individui non viene creata:

1. Attraverso una funzione, tipicamente crescente della fitness, viene selezionata una coppia di cromosomi, che diventeranno i genitori, mediante l'assegnazione di una certa probabilità nella selezione. Inoltre possono essere selezionati più volte durante la fase di generazione di una nuova popolazione.

2. Definita la probabilità di cross-over p_c, viene effettuata tale operazione sui cromosomi precedentemente selezionati in un punto scelto casualmente, generando dunque due figli. Nel caso in cui il cross-over non avvenisse, la stringa viene semplicemente copiata.

3. Si effettua la mutazione in ogni locus con probabilità p_m

Vengono quindi generate le nuove popolazioni finché non verrà soddisfatto un certo criterio di convergenza.
Anche se non esiste una teoria matematica rigorosa dietro l'utilizzo degli algoritmi genetici esiste però un teorema, noto come *teorema di Holland* o *teorema degli schemi* che assicura la convergenza di un GA verso una soluzione ottimale.
Non tutti i bit di una stringa contribuiscono alla soluzione ottimale. All'interno di un cromosoma è infatti possibile identificare sotto-stringhe, definite appunto come *schemi*. Ad esempio, sia un generico individuo codificato come

$$< 0111000 >$$

Un possibile schema potrebbe essere definito come

$$H =< *11***0 >$$

Questi blocchi, se ricombinati in modo adeguato e dunque se fatti sopravvivere, permettono di ottenere una soluzione ottimale. In tal caso, vengono definiti *schemi costruttori* [19].
Si definisce *ordine di uno schema* $o(H)$ il numero di posizione fisse di uno schema mentre *la lunghezza di definizione* $\delta(H)$ è la distanza tra il primo e l'ultimo bit specifico. Per cui, è possibile definire come *fitness* di uno schema, la media di tutte le fitness ricavate dalle stringhe che corrispondono ad un determinato schema.
Il teorema afferma che gli schemi di basso ordine caratterizzati da una fitness elevata aumentano in maniera esponenziale nelle generazioni successive [20]

$$W(m(H,t+1)) \geq \frac{m(H,t)f(H)}{a_t}[1-p] \tag{1.2}$$

dove $m(H,t+1)$ è il numero di stringhe appartenenti allo schema H nella generazione t, p la probabilità di interruzione, $f(H)$ è la fitness media osservata allo schema H ed a_t è la fitness media osservata alla generazione t.
Non è però detto che da due genitori aventi una fitness elevata si ottengano dei

figli caratterizzati da una fitness pressoché simile o maggiore. Ciò che invece si può dire è che passando i migliori schemi alla generazione successiva, aumenta la probabilità di trovare soluzioni migliori.
In particolare, la probabilità di interruzione p ovvero la probabilità che il cross-over o la mutazione distruggano lo schema H è definita come

$$p = \frac{\delta(H)}{l-1} p_c + o(H) p_m \tag{1.3}$$

dove l è la lunghezza della stringa, p_m la probabilità di mutazione e p_c la probabilità di cross-over. Ciò che si può osservare è che uno schema avente una lunghezza di definizione $\delta(H)$ piccola ha meno probabilità di essere interrotto.

1.5 Particle Swarm Optimization

Il PSO, Particle Swarm Optimization, è un algoritmo appartenente alla famiglia degli algoritmi *population-based* [21] i quali si basano su tre principali paradigmi

1. Una popolazione di individui è utilizzata all'interno del processo di ricerca
2. L'informazione sulla fitness è direttamente usata nella ricerca
3. Le leggi, siano esse deterministiche o probabilistiche, che regolano l'aggiornamento delle soluzioni migliorano la qualità della soluzione

Il principale intento di tale metodologia di ottimizzazione era quello di simulare graficamente l'imprevedibile movimento di uno stormo di uccelli con l'obiettivo di individuare possibili modelli capaci di governare la capacità degli individui di volare in modo sincrono o di cambiare improvvisamente le posizioni relative al fine di creare un raggruppamento ottimale. Da ciò, l'idea evolvé diventando dunque un semplice ed efficiente algoritmo di ottimizzazione [22].
La caratteristica principale del PSO è l'interazione sociale. Infatti, gli individui o *particelle*, "volano" all'interno dello spazio di ricerca in una ben determinata struttura sociale, cambiando posizione basandosi sulla tendenza social-psicologica degli individui di emulare il successo dei loro simili. Tale modifica è quindi influenzata dall'esperienza o conoscenza degli individui della popolazione con la conseguenza che nel processo di ricerca le particelle ritornano stocasticamente nei pressi della precedente regione all'interno della quale è stata individuata una buona soluzione.
All'interno di una rete altamente connessa l'informazione inerente ad una buona soluzione viaggia più velocemente rispetto ad una struttura più indipendente. In termini di ottimizzazione, ciò consente una miglior convergenza al risultato ottimale rispetto ad una rete meno connessa.

Figura 1.5: Esempio di uno stormo di uccelli

Tuttavia tale tipologia di reti risulta essere suscettibile ai punti di minimo locale a causa della robusta trasmissione dell'informazione che chiaramente richiama gli individui ad avvicinarsi verso la soluzione individuata.
Di contro, nelle reti sparse caratterizzate da un certo numero di raggruppamenti può succedere che non vi sia una adeguata copertura del dominio. Infatti, ciascun raggruppamento copre soltanto una piccola parte dello spazio di ricerca e sono inoltre caratterizzate da una bassa connettività. Di conseguenza, l'informazione riguardo una parte ristretta dello spazio di ricerca è condivisa sotto forma di un flusso di informazioni limitato tra i vari raggruppamenti.

1.5.1 Reti sociali

In particolare, per il PSO sono state sviluppate diverse reti sociali [22]. Vengono di seguito riportate le principali:

1. Nella struttura a *stella* (immagine *a* in 1.6), tutte le particelle sono interconnesse fra di loro. Ognuna di esse può dunque comunicare con qualsiasi altra particella appartenente allo stormo e tutte sono portate ad imitare la particella che rappresenta la migliore soluzione individuata. Aspetto negativo di tale topologia è la tendenza a rimanere intrappolati in punti di minimo locale.
   Il PSO che utilizza tale rete sociale viene definito come $gbestPSO$.

2. Nella struttura ad *anello* (immagine *b* in 1.6) ogni particella comunica con le N particelle immediatamente a lei vicine. Per cui, ogni particella tenta

di imitare quella migliore del raggruppamento alla quale appartiene. Inoltre, all'interno dello stormo tali raggruppamenti possono sovrapporsi facilitando lo scambio di informazioni (performance migliori rispetto la struttura a stella) con però una convergenza più lenta.
Il PSO che utilizza tale rete sociale viene definito come $lbestPSO$.

3. Nella struttura a *ruota* (immagine c in 1.6) una particella ha il compito di diffondere le informazioni provenienti dalle altre particelle del raggruppamento. In particolare, viene effettuato un processo di comparazione in seguito al quale la propria posizione è aggiornata in funzione di quella migliore. Se l'aggiornamento risulta essere positivo, ovvero la particella principale si sposta in una posizione migliore della precedente, allora avviene la comunicazione con le altre particelle.

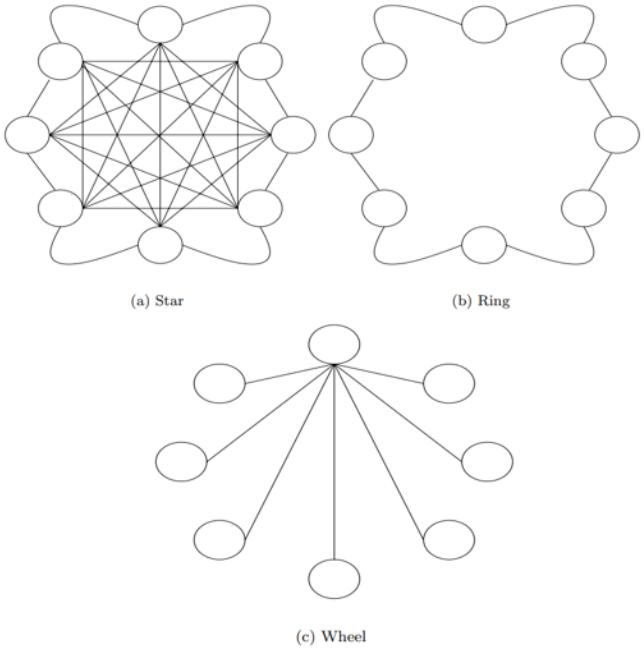

Figura 1.6: Tipiche reti sociali [22]

1.5.2 L'algoritmo

Sia N la dimensione dello stormo, ogni particella i può essere rappresentata come un oggetto avente 3 differenti caratteristiche

1. $\mathbf{x_i}$: la posizione attuale della particella

2. $\mathbf{v_i}$: la velocità corrente della particella

3. $\mathbf{y_i}$: la migliore posizione di sempre individuata dalla particella

L'aggiornamento della posizione del vettore $\mathbf{y_i}$ alla iterazione t avviene secondo l'Equazione 3.19

$$\mathbf{y_i}(\mathbf{t+1}) = \begin{cases} \mathbf{y_i}(\mathbf{t}) & \text{se } f(\mathbf{x_i}(\mathbf{t+1})) \geq f(\mathbf{y_i}(\mathbf{t})) \\ \mathbf{x_i}(\mathbf{t+1}) & \text{se } f(\mathbf{x_i}(\mathbf{t+1})) < f(\mathbf{y_i}(\mathbf{t})) \end{cases} \tag{1.4}$$

In particolare, l'aggiornamento della posizione avviene mediante l'utilizzo di una velocità $\mathbf{v_i}(\mathbf{t})$, in accordo all'Equazione 1.5

$$\mathbf{x_i}(\mathbf{t+1}) = \mathbf{x_i}(\mathbf{t}) + \mathbf{v_i}(\mathbf{t+1}) \tag{1.5}$$

E' proprio il vettore velocità $\mathbf{v_i}(t+1)$ che guida il processo di ottimizzazione e contiene in se sia la conoscenza dettata dalla esperienza di ogni singola particella, nota come *componente cognitiva* e proporzionale alla distanza tra la posizione di una generica particella e quella inerente alla propria migliore posizione individuata, sia quella inerente allo scambio di informazioni all'interno della popolazione che cade sotto il nome di *componente sociale*.
In particolare, come anticipato nel paragrafo 1.5.1, sono due le versione del PSO: il *lbestPSO* ed il *gbestPSO*. La differenza tra i due algoritmi risiede nel set di particelle con il quale un dato individuo interagisce.
Nel *gbestPSO* esiste un solo raggruppamento, ovvero l'intero stormo, che sfrutta una topologia a stella. L'informazione sociale è dunque rappresentata dalla miglior posizione individuata dallo stormo che verrà indicata come $\hat{\mathbf{y}}(t)$

$$\hat{\mathbf{y}}(t) \in \{\mathbf{y_0}(t), \ldots, \mathbf{y_{n_s}}(t)\} | f(\hat{\mathbf{y}}(t)) = min\{f(\mathbf{y_0}(t)), \ldots, f(\mathbf{y}_{n_s}(t))\} \tag{1.6}$$

dove n_s è il numero totale delle particelle dello stormo.
Il *lbestPSO* utilizza invece una topologia ad anello, nella quale piccoli raggruppamenti sono definiti per ogni particella. In questo caso, la miglior particella $\hat{\mathbf{y}}_i(t)$ rappresenta la migliore posizione all'interno del raggruppamento $\mathcal{N}_i$

$$\hat{\mathbf{y}}_i(t) \in \mathcal{N}_i | f(\hat{\mathbf{y}}(t)) = min\{f(\mathbf{a})\}, \quad \forall \mathbf{a} \in \mathcal{N}_i \tag{1.7}$$

Tipicamente il *lbestPSO* è preferito in quanto favorisce la diffusione delle informazioni grazie alla sovrapposizione dei raggruppamenti, evitando dunque di rimanere intrappolato in un punto di minimo locale. E' possibile però migliorare le prestazioni del *gbestPSO* modificando opportunamente alcuni parametri che verranno successivamente introdotti ed introducendo alcune modifiche alla versione originale

del codice: in questo modo si riesce dunque a mantenere una velocità di convergenza maggiore rispetto alla versione locale dell'algoritmo eliminando al contempo il fenomeno della *stagnation* ovvero di intrappolamento in un punto di minimo locale.

A prescindere dalla natura dell'algoritmo, è possibile individuare un corpo base di quest'ultimo, riassunto di seguito

Algorithm 1 Algoritmo PSO

```
▷ Creare ed inizializzare uno stormo di dimensione n_x
repeat
    for ogni particella i=1,..., n_s do
        if f(x_i) < f(y_i) then
            y_i ← x_i
        end if
        if f(y_i) < f(ŷ_i) then
            ŷ_i ← y_i
        end if
    end for
    for ogni particella i=1,..., n_s do
        ▷ Aggiornare la velocità di ogni particella
        ▷ Aggiornare la posizione di ogni particella
    end for
until condizione di convergenza soddifsfatta
```

Ulteriore aspetto da valutare è la scelta del criterio di convergenza. In letteratura è possibile individuare le seguenti metodologie

1. La ricerca termina quando viene raggiunto un numero massimo di iterazioni. Chiaramente, se il numero delle iterazioni è basso vi è maggiore probabilità di convergere in un punto che non è di minimo globale. Per cui è tipicamente usato insieme ad un altro criterio che forza il blocco della ricerca quando l'algoritmo fallisce nella convergenza.

2. La ricerca termina quando una soluzione accettabile è stata individuata. Sia $\mathbf{x}^*$ l'ottimo della funzione obiettivo f, tale condizione è soddisfatta quando

$$f(\mathbf{x}_i) \leq |f(\mathbf{x}^* - \epsilon)| \quad (1.8)$$

   dove il valore del parametro di soglia ϵ va individuato con cura: se troppo grande, la ricerca termina con una soluzione non ottimale. Di contro, la ricerca potrebbe non terminare e continuare all'infinito.

3. La ricerca termina quando non sono osservati miglioramenti per un certo numero di iterazioni. Ad esempio, se il cambio di posizioni della particella è

relativamente piccolo allora si può considerare che lo stormo abbia converso in un minimo. Oppure, se la velocità risulta essere pressoché nulla per un certo numero di iterazioni, la ricerca può ritenersi conclusa. In fine, la ricerca può terminare anche se non si osservano miglioramenti significativi entro un certo numero di cicli.
Vanno dunque definiti due parametri: il primo inerente alla finestra di iterazioni da considerare per stoppare la ricerca, il secondo relativo ad una soglia per considerare accettabile o inaccettabile la performance.

4. La ricerca termina quando il raggio normalizzato dello stormo è prossimo a zero, ovvero quando R_{norm} è inferiore ad una certa soglia. Si definisce raggio normalizzato la seguente quantità

$$R_{norm} = \frac{R_{max}}{diameter(S)} \tag{1.9}$$

dove $diameter(S)$ è il diametro dello stormo iniziale e R_{max} è definito come

$$R_{max} = ||\mathbf{x}_m - \hat{\mathbf{y}}|| \geq ||\mathbf{x}_i - \hat{\mathbf{y}}||, \qquad m = 1, \ldots, n_s \wedge \forall i = 1, \ldots, n_s \tag{1.10}$$

Se R_{norm} è vicino a zero, lo stormo ha poche probabilità per migliorare la soluzione a meno che la migliore soluzione non stia ancora cambiando posizione.

5. La ricerca termina quando la pendenza della funzione obiettivo è approssimativamente nulla

$$f'(t) = \frac{f(\hat{\mathbf{y}}(t)) - f(\hat{\mathbf{y}}(t-1))}{\hat{\mathbf{y}}(t) - \hat{\mathbf{y}}(t-1)} < \epsilon \tag{1.11}$$

Per cui, se $f'(t)<\epsilon$ per un certo numero di iterazioni, lo stormo ha converso. Tale criterio di convergenza risulta essere superiore ai metodi prima citati in quanto si determina se lo stormo stia ancora effettuando progressi usando le informazioni dello spazio di ricerca.
Un aspetto negativo risiede nel fatto che la ricerca termina anche quando alcune delle particelle vengono attratte in un punto di minimo locale, non dando la possibilità a quelle particelle coinvolte nella fase di esplorazione di poter individuare una soluzione migliore. Per risolvere questo problema, il metodo della pendenza può essere usato insieme al metodo del raggio dello stormo, per fornire l'indicazione della convergenza di tutte le particelle in un punto.

1.5.3 Modello del comportamento sociale

Verrà di seguito analizzato il modello inerente al comportamento sociale, ovvero all'aggiornamento delle velocità di ogni particella del *gbest* PSO essendo poi uno dei

due algoritmi implementati per ottimizzare la collocazione dei sensori sul modello in scala dell'ala testata in laboratorio.
In particolare la velocità della i-esima particella è calcolata in accordo alla seguente equazione

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}[\hat{y}_j(t) - x_{ij}(t)] \tag{1.12}$$

dove $v_{ij}(t)$ è la velocità della i-esima particella nella dimensione $j = 1, \ldots, n_x$ all'iterazione t, $x_{ij}(t)$ è la posizione della i-esima particella nella dimensione $j = 1, \ldots, n_x$ all'iterazione t, c_1 e c_2 sono dei parametri di accelerazione usati per scalare il contributo della componente cognitiva e sociale mentre $r_{1j}(t)$ e $r_{2j}(t)$ sono dei valori casuali campionati da una distribuzione uniforme all'interno dell'intervallo [0,1]: questi due ultimi valori introducono un aspetto stocastico nell'algoritmo. Analizzando nel dettaglio l'equazione 1.12 è possibile individuare

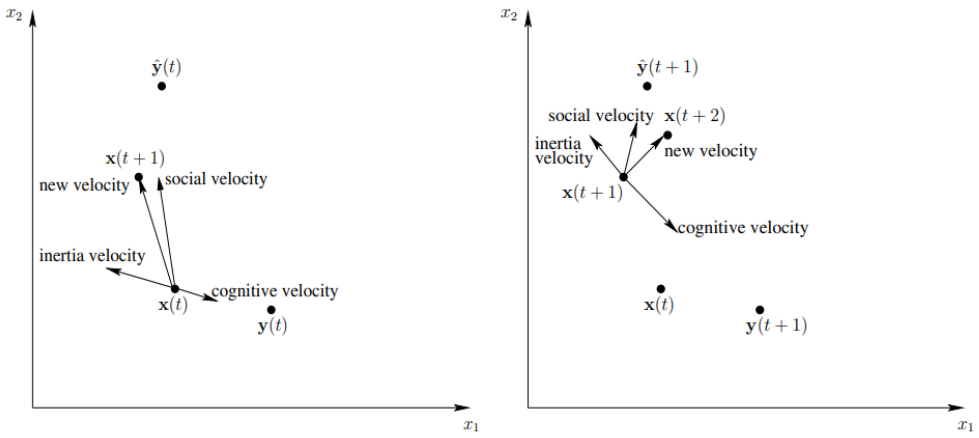

Figura 1.7: Illustrazione geometrica dell'aggiornamento delle velocità e delle posizioni per una particella bi-dimensionale [22]

1. La componente sociale, $c_2 r_{2j}[\hat{y}_j(t) - x_{ij}(t)]$ che quantifica le performance della i-esima particella relativamente al gruppo di particelle. Concettualmente, tale componente rappresenta uno standard di gruppo che gli individui cercano di raggiungere: ogni particella tende a spostarsi verso la migliore posizione individuata. Viene inoltre definita come *invidia*.

2. La componente cognitiva, $c_1 r_{1j}[y_{ij}(t) - x_{ij}(t)]$ che quantifica la performance della i-esima particella relativamente al proprio passato. Rappresenta la memoria individuale inerente alla migliore propria migliore posizione individuata. L'effetto di tale componente è quello di far muovere la particella verso la propria migliore posizione individuata. Viene inoltre definita come *nostalgia*.

3. La velocità precedente $v_{ij}(t)$ rappresenta invece una sorta di memoria del volo precedente, relativa dunque al passato immediato. Tale componente può essere interpretata come una sorta di inerzia che ostacola il repentino cambio di direzione nella nuova iterazione.

In Figura 1.7 è possibile osservare una rappresentazione geometrica delle velocità. I coefficienti c_1 e c_2 vengono definiti coefficienti di accelerazione. Se ad esempio c_1=c_2=0, le particelle volano a velocità costante finché non incontrano i limiti del dominio di ricerca. Se c_1>0 e c_2=0, tutte le particelle sono indipendenti fra di loro. Ognuna di esse trova la propria migliore posizione, senza comunicare alle altre particelle tale informazione: viene quindi eseguita una ricerca di tipo locale. Invece, se c_2>0 e c_1=0, tutto lo stormo è attratto dalla migliore posizine $\hat{\mathbf{y}}$.
Ciò che si osserva è che la ricerca risulta ottimizzata se c_1=c_2 in quanto le particelle saranno attratte sia dalla propria migliore posizione $\mathbf{y_i}$ che da quella globale $\hat{\mathbf{y}}$. Anche se in molte applicazioni, viene tipicamente effettuata tale scelta, in realtà è possibile stabilire delle leggi che regolano l'andamento dei coefficienti di accelerazione durante il susseguirsi delle iterazioni.
Chiaramente, se $c_1 \gg c_2$ ogni particella sarà maggiormente attratta dalla propria migliore posizione, invece, se $c_2 \gg c_1$, tenderà ad andare incontro alla migliore fra tutte. Un possibile schema adattivo per la valutazione di c_1 e c_2 è il seguente

$$c(t) = \frac{c_{min} + c_{max}}{2} + \frac{c_{max} - c_{min}}{2} + \frac{e^{-m_i(t)} - 1}{e^{-m_i(t)} + 1} \tag{1.13}$$

dove il parametro $m_i(t)$, noto come *miglioramento relativo*, è definito come

$$m_i(t) = \frac{f(\hat{\mathbf{y}}_\mathbf{i}(t)) - f(\mathbf{x_i}(t))}{f(\hat{\mathbf{y}}_\mathbf{i}(t)) + f(\mathbf{x_i}(t))} \tag{1.14}$$

e rappresenta la pendenza dello spazio di ricerca nella posizione di ogni particella. Un'altro approccio è quello di far incrementare linearmente c_2 e di far diminuire nel medesimo modo c_1. Tale strategia si concentra dunque sull'esplorazione nelle prime fasi della ricerca mentre punta a definire un miglior minimo, ricercando nei dintorni della migliore soluzione individuata durante le ultime iterazioni. Le leggi che regolano l'andamento dei due coefficienti sono dunque le seguenti

$$c_1(t) = (c_{1,min} - c_{1,max})\frac{t}{n_t} + c_{1,max} \tag{1.15}$$

$$c_2(t) = (c_{2,min} - c_{2,max})\frac{t}{n_t} + c_{2,max} \tag{1.16}$$

dove tipicamente $c_{1,max}$=$c_{2,max}$=2.5 e $c_{1,min}$=$c_{2,min}$=0.5.
Un importante aspetto che determina l'efficienza e l'accuratezza di un algoritmo

di ottimizzazione è il trade off tra fase di *exploration* ed *exploitation*. La fase di exploration è dettata dall'abilità dell'algoritmo di esplorare differenti regioni dello spazio di ricerca in modo da individuare un punto di minimo. La fase di *exploitation*, invece, rappresenta la capacità di concentrare la ricerca in prossimità di un'area in modo da rifinire la soluzione individuata. Per avere un giusto bilanciamento tra le due fasi viene quindi introdotto un parametro d'inerzia che premoltiplica la velocità $v_{ij}(t)$ della particella

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}[\hat{y}_j(t) - x_{ij}(t)] \tag{1.17}$$

Per w ≥ 1, la velocità aumenta ad ogni iterazione, accelerando la particella fino ad una v_{max} definibile e facendo divergere lo stormo: le particelle dunque non riescono a variare la propria direzione di volo. Per w < 1, le particelle decelerano fino a raggiungere una velocità nulla. In particolare, valori elevati di w aumentano la capacità esplorativa dello stormo, di contro, valori bassi, favoriscono la fase di *explotation* riducendo a minimo quella di *exploration*.
Anche in questo caso è possibile definire un parametro di inerzia variabile ad ogni iterazione. Tipicamente, si parte da un w(0) elevato per poi arrivare, alle ultime iterazioni, ad un valore relativamente basso. Così facendo, anche in questo caso viene favorita la fase di esplorazione durante le prime iterazioni per poi concludere la ricerca, migliorando la soluzione individuata.
Diversi sono gli approcci, di seguito riportati

1. *Random*: il parametro di inerzia viene selezionato in maniera random da una distribuzione di tipo gaussiana

2. *Decrescita lineare*: si parte da un valore iniziale w(0)=0.9 per arrivare all'ultima iterazione ad un valore pari a w(n_t)=0.4

$$w(t) = (w(0) - w(n_t))\frac{n_t - t}{n_t} + w(n_t) \tag{1.18}$$

   Tale tipologia di decrescita tende a favorire la fase di esplorazione

3. *Decrescita non lineare*: viene seguito un andamento non lineare che riduce la fase di ricerca, dando maggior importanza all'exploitation. Tra le leggi più utilizzate troviamo la seguente

$$w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n_t + 0.4} \tag{1.19}$$

Differenti studi teorici hanno dimostrato come la convergenza del PSO è influenzata dai valori dei coefficienti di inerzia e di accelerazione.
Si dimostra che la traiettoria di una particella converge se

$$1 > w > \frac{1}{2}(\phi_1 + \phi_2) - 1 \geq 0, \qquad w \in [0,1)] \tag{1.20}$$

dove ϕ_1=$c_1\mathcal{U}(0,1)$ e ϕ_2=$c_2\mathcal{U}(0,1)$. Per cui, dato che c_1 e c_2 rappresentano gli estremi di ϕ_1 e ϕ_2 è possibile riscrivere l'equazione 1.20 nel seguente modo

$$1 > w > \frac{1}{2}(c_1 + c_2) - 1 \geq 0, \qquad w \in [0,1] \tag{1.21}$$

L'equazione 1.21 è valida però solo nel caso in cui i parametri stocastici r_{1j} e r_{2j} non vengano utilizzati. In questo caso, la convergenza può essere osservata se il parametro ϕ_{ratio}, definito come

$$\phi_{ratio} = \frac{\phi_{crit}}{c_1 + c_2} \tag{1.22}$$

è prossimo al valore unitario, dove

$$\phi_{crit} = \sup \phi \mid 0.5\phi - 1 < w, \qquad w \in (0, c_1 + c_2] \tag{1.23}$$

Ulteriore aspetto importante è l'inizializzazione delle velocità delle particelle [23]. Mentre la posizione delle particelle è inizializzata in maniera randomica all'interno del dominio, è possibile seguire differenti approcci per la velocità

1. La velocità può essere inizializzata a 0 per tutte le particelle, $\mathbf{v}_i(0) = \mathbf{0}$ con i=$1, \ldots, n_s$. Una criticità di quest'approccio risiede nel fatto di limitare l'iniziale capacità di esplorazione dello stormo e quindi la copertura iniziale dello spazio di ricerca.

2. La velocità può essere inizializzata in maniera randomica, campionando all'interno dell'intervallo (-x_{min}, x_{max})^{n_x}. Tale metodologia offre il vantaggio di favorire la fase di esplorazione, iniziale, fornendo alle particelle movimenti più ampi. Però, così come larghi step iniziali aumentano la diversità dello stormo, incrementano anche la possibilità di violare i limiti dello spazio di ricerca.
   Una possibile correzione, è quella di inizializzare la velocità con valori casuali piccoli.

Tipicamente è preferibile inizializzare la velocità a valori nulli oppure usare bassi valori randomici.

1.5.4 Guaranteed Convergence PSO

Il modello base del PSO presenta una criticità: quando $\mathbf{x_i}$=$\mathbf{y_i}$=$\hat{\mathbf{y}}$, l'aggiornamento delle velocità dipende solo dal termine $w\mathbf{v_i}$. Se questa condizione fosse verificata per tutte le particelle e se dovesse persistere per un certo numero di iterazioni, allora il contributo $w\mathbf{v_i}$ tenderebbe a 0, facendo dunque convergere tutte le particelle in un unico punto. Tale problematica è nota come *stagnation*. Per evitare questo, il

PSO può essere forzato, variando la miglior posizione $\hat{\mathbf{y}}$: viene quindi di seguito introdotto il GCPSO.
Sia τ l'indice della miglior particella

$$\mathbf{y}_\tau = \hat{\mathbf{y}} \tag{1.24}$$

Il GCPSO, Guaranteed Convergence PSO, cambia la posizione di questa particella come

$$x_{\tau j}(t+1) = \hat{y}_j(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_2(t)) \tag{1.25}$$

dove l'aggiornamento della velocità è ottenuto in accordo alla seguente equazione

$$v_{\tau j}(t+1) = -x_{\tau j}(t) + \hat{y}_j(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_{2j}(t)) \tag{1.26}$$

dove $\rho(t)$ è un fattore di scala, definito successivamente. Tale equazione è però solo utilizzata per variare la posizione della miglior particella mentre per tutte le altre particelle rimane ancora valida l'Equazione 1.17.
In particolare, il termine ρ(t)=$(1 - 2r_{2j}(t))$ genera un campionamento randomico all'interno di uno spazio di campionamento di dimensione $2\rho(t)$. In questo modo, si forza il PSO ad eseguire una ricerca randomica in un'area circostante alla posizione $\hat{\mathbf{y}}$. Il parametro $\rho(t)$, che controlla il diametro di tale area, è aggiornato in accordo alla seguente legge

$$\begin{cases} 2\rho(t) & \text{se } \#\text{successes(t)} > \epsilon_s \\ 0.5\rho(t) & \text{se } \#\text{failures(t)} > \epsilon_f \\ \rho(t) & \text{altrimenti} \end{cases} \tag{1.27}$$

dove $\#success$ e $\#failures$ rappresentano il numero di successi e di failures consecutivi. Una failure è definita se $f(\hat{\mathbf{y}}(t)) \leq f(\hat{\mathbf{y}}(t+1))$, ovvero

$$\#successes(t+1) > \#successes(t) \rightarrow \#failures(t+1) = 0$$

$$\#failures(t+1) > \#failures(t) \rightarrow \#failures(t+1) = 0$$

Per quanto riguarda la scelta dei parametri soglia un tipico valore raccomandato è di 15 per ϵ_s e di 5 per ϵ_f, anche se il valore può cambiare a seconda del problema. Inoltre un valore di ρ dinamico può essere utilizzato al fine di individuare la migliore dimensione per il campionamento, dato un certo stato dell'algoritmo. Quando la miglior posizione è ripetutamente migliorata per uno specifico valore di ρ, quest'ultimo viene aumentato per permettere nelle iterazioni successive di aumentare lo spazio di ricerca.

1.5.5 Binary PSO

Il PSO generico è stato sviluppato per risolvere problemi di ottimizzazione continua basati su due premesse: l'utilizzo della distanza euclidea per valutare la distanza tra due particelle e la continuità della funzione obiettivo.
Quando il dominio della funzione obiettivo è però discreto ed inoltre risulta anche essere finito, si parla invece di *ottimizzazione combinatoria* ed alcune modifiche, alla struttura originale dell'algoritmo, devono essere introdotte.
La principale variazione è l'introduzione del concetto di *distanza di Hamming* che si riferisce al numero di punti in cui due linee di codice binario differiscono: ad esempio, la distanza tra i due numeri binari 10101 e 00100 risulta essere due, dal momento che differiscono per due bit (chiaramente in posizioni diverse). Dato che la distanza di Hamming è un metodo di misura discreto, il PSO che si basa su tale definizione risulta essere per natura discreto, senza necessità di introdurre alcuna operazione di arrotondamento.
In particolare, in uno spazio binario, ciascuna particella è codificata con un certo numero di bit. Ad una determinata sequenza corrisponde un vertice di quello che potrebbe essere definito come *ipercubo*: per cui, al variare di alcune cifre binarie, la particella cambia posizione, ovvero il vertice nel quale risulta essere collocata [24].
Ulteriore cambio di paradigma è il significato che viene dato alla velocità di ogni particella: nel BPSO, Binary PSO, la velocità assume un significato probabilistico, ovvero, la *probabilità di cambiamento* di un bit. Ad esempio, sia la componente d-esima della i-esima particella pari a $x_{id} = 0$. Se v_{id}=0.20, allora si ha il venti per cento di probabilità che essa rimanga pari a zero e l'ottanta per cento di probabilità che essa cambi ad uno.
Per cui, l'equazione 1.17 rimane la medesima con la differenza che le posizioni $\mathbf{x}_i$, $\mathbf{y}_i$ e $\hat{\mathbf{y}}$ sono ora codificate in binario e che la velocità assume un significato di probabilità.
Inoltre, viene introdotta una funzione logica di trasformazione $\mathcal{S}(v_{id})$. La logica del cambio della cifra binaria è la seguente

$$\begin{aligned} &if\ \ (rand() < \mathcal{S}(v_{id}))\ \ then\ \ x_{id} = 1 \\ &else\ \ x_{id} = 0 \end{aligned}$$

dove $\mathcal{S}(v_{id})$ rappresenta la funzione sigmoidea rappresentata in Figura 1.8

$$\mathcal{S}(v_{id}) = \frac{1}{1 + exp(-v_{id})} \tag{1.28}$$

che chiaramente fornisce un valore compreso all'interno dell'intervallo [0.0, 1.0] da confrontare con un valore casuale campionato da una distribuzione uniforme all'interno del medesimo intervallo.
Viene inoltre introdotta una velocità massima V_{max} che va a limitare il valore delle

probabilità. Si supponga di porre V_{max}=6, dato che $|v_{id}|<V_{max}$ allora il valore di probabilità sarà compreso nel nuovo intervallo [0.0025, 0.9975]. Il risultato di tale limitazione è l'introduzione di un minor tasso di variabilità all'interno della popolazione: se nella versione continua del PSO a valori alti di tale parametro corrisponde un incremento dello spazio scansionato durante la fase di exploration, nel BPSO il risultato è l'esatto contrario per quanto precedentemente detto.

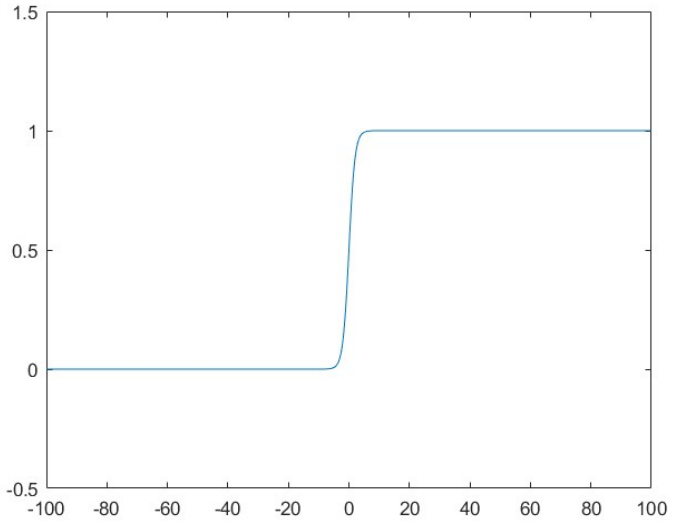

Figura 1.8: Funzione sigmoidea

A questo punto pare opportuno cercare di introdurre un interpretazione alla ricerca, che seppur di natura stocastica, utilizza sempre l'equazione 1.17 per aggiornare i valori delle velocità v_{id}. Nel PSO infatti ciascuna componente della suddetta equazione ha un proprio significato di ispirazione naturale, rifacendosi infatti al comportamento biologico degli individui di uno stormo. Nel BPSO le traiettorie sono puramente stocastiche e per di più compiute all'interno di uno spazio di ricerca multidimensionale assimilabile ad un ipercubo. Ciò che si può osservare è che se anche la componente del vettore velocità v_{id} rimane fissa, la posizione della particella x_{id} potrebbe ugualmente cambiare attraverso il cambio del singolo bit.
In particolare, la probabilità che un bit diventi uno sarà pari a $\mathcal{S}(v_{id})$ mentre la probabilità che esso diventi zero sarà 1-$\mathcal{S}(v_{id})$. Per giunta, se invece il bit di partenza fosse zero, le probabilità chiaramente si invertirebbero. Quindi, è possibile conferire una nuova interpretazione al cambio di v_{id} di tipo *non-direzionale* contro il cambio, o meglio dire alterazione di natura direzionale, nella v_{id} della versione continua del codice.
Si ponga ora l'attenzione sulla funzione sigmoidea $\mathcal{S}(v_{id})$. In generale, essa permette

di mappare valori continui in valori discreti: tale tipologia di funzione prende il nome di *funzione di trasferimento* e forza gli agenti di ricerca a muoversi all'interno di uno spazio binario. Caratteristica di tali funzioni [25] è quella di fornire un'alta probabilità nel cambio della posizione per un intervallo abbastanza grande di velocità ed una piccola probabilità per un intervallo limitato. In aggiunta, il codominio di una funzione di trasferimento dovrebbe essere pari all'intervallo [0,1]. E' possibile classificare le funzioni di trasferimento in due principali famiglie, *s-shape* e $v - shape$, riportate nella Tabella 1.2

#	Funzione di trasferimento	Famiglia
1	$\mathcal{S}(v_{id}) = \frac{1}{1+e^{-2v_{id}}}$	s-shape
2	$\mathcal{S}(v_{id}) = \frac{1}{1+e^{-v_{id}}}$	s-shape
3	$\mathcal{S}(v_{id}) = \frac{1}{1+e^{-\frac{v_{id}}{2}}}$	s-shape
4	$\mathcal{S}(v_{id}) = \frac{1}{1+e^{-\frac{v_{id}}{3}}}$	s-shape
5	$\mathcal{S}(v_{id}) = erf(\frac{\sqrt{\pi}}{2}v_{id})$	v-shape
6	$\mathcal{S}(v_{id}) = \lvert tanh(v_{id})\rvert$	v-shape
7	$\mathcal{S}(v_{id}) = \lvert\frac{v_{id}}{\sqrt{1+v_{id}^2}}\rvert$	v-shape
8	$\mathcal{S}(v_{id}) = \lvert\frac{2}{\pi}arctan(\frac{\pi}{2}v_{id})\rvert$	v-shape

Tabella 1.2: Funzioni di trasferimento

Vengono in Figure 1.9 rappresentate le due tipologie di funzioni.
Se con le funzioni di trasferimento appartenenti alla famiglia s-shape viene utilizzata la logica di aggiornamento della posizione precedentemente riportata, con quelle della famiglia v-shape è invece utilizzata la seguente regola

$$if\ \ (rand() < \mathcal{S}(v_{id}))\ \ then\ \ x_{id} = complement(x_{id})$$
$$else\ \ x_{id} = x_{id}$$

Ciò che si evince in [25] è che in alcune funzioni obiettivo di benchmark i risultati ottenuti con le funzioni di trasferimento v-shape risultano migliori in termini di performance. In particolar modo, l'algoritmo presenta una miglior capacità di evitare punti di minimo locale, una maggior velocità di convergenza ed una migliore accuratezza nel risultato finale.

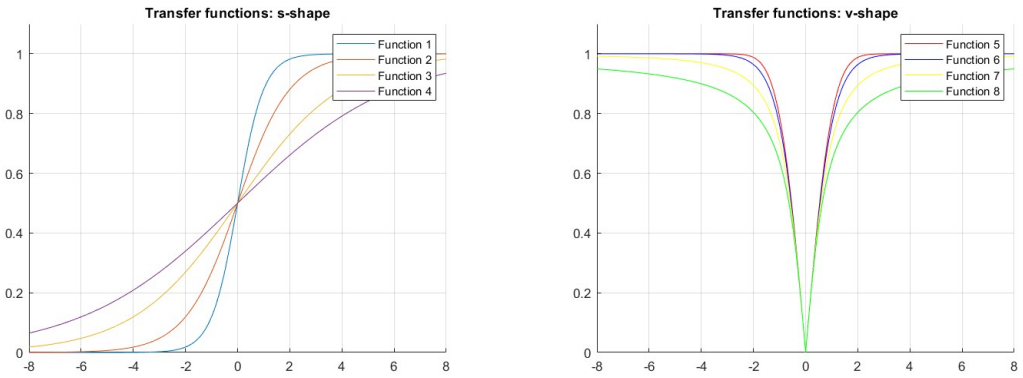

Figura 1.9: Funzioni di trasferimento: s-shape e v-shape

1.6 Whale Optimization Algorithm

Le megattere sono cetacei diffusi negli oceani e nei mari di tutto il mondo e secondo alcuni dati del 2018, sono 84.000 gli individui esistenti. Il nome deriva dai termini greci "méga" e "pterón", ovvero "grande ala".
Questi animali vengono generalmente avvistati in piccoli gruppi che rimangono uniti per pochi giorni o settimane al massimo, ad eccezione delle coppie madre e figlio, sebbene gli esemplari che vanno a caccia e i maschi che competono tra loro per le femmine possano formare anche grandi aggregazioni. In particolare, durante la stagione degli amori i maschi di megattera producono i suoni, anche dette canzoni, più lunghi e complessi. Le megattere, infatti, hanno una complessa comunicazione sonora che si differenzia non solo da individuo a individuo ma anche in base alla vicinanza della stagione riproduttiva [26] e grazie alla quale è possibile l'interazione anche con altre specie di cetacei. Sono inoltre estremamente attive in superficie ed esibiscono un'intera serie di comportamenti aerei, ad esempio saltando (breaching) e battendo la superficie con la coda (lobtailing) e le pinne pettorali. Tali attività potrebbero costituire una particolare forma di gioco, di comunicazione o un sistema per rimuovere i parassiti [27].
Le megattere sono esseri molto intelligenti capaci di pensare, comunicare, imparare ed anche emozionarsi: in particolare, il cervello di questi mammiferi contiene il

triplo di cellule fusiformi rispetto a quello dell'uomo le quali sono responsabili delle capacità cognitive, dell'empatia e dell'organizzazione sociale. Inoltre, riescono anche a formare reti sociali abbastanza complesse all'interno delle quali vige una comunicazione strutturata.
Hanno un'alimentazione generalista e la loro dieta è costituita soprattutto da krill e piccoli pesci che si spostano in banco. L'aspetto più interessante è la strategia di caccia adottata definita *bubble-net foraging method* ed osservata per la prima volta nel 2011 [28].

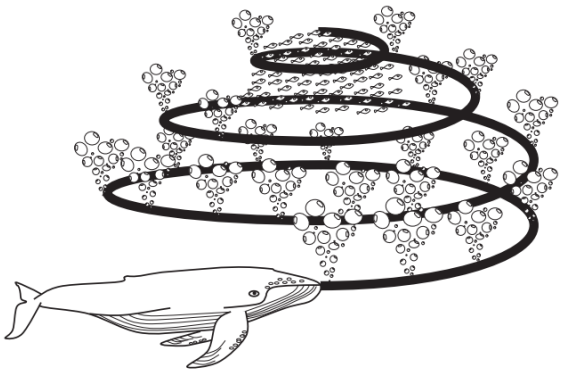

Figura 1.10: Bubble-net feeding behavior [28]

Sono presenti tipicamente due principali manovre dettate dalla non elevata velocità alla quale le balenottere possono spingersi. Nella prima, chiamata *upward-spirals*, le balene si immergono per 12 metri ed iniziano a creare delle bolle lungo una traiettoria a spirale salendo verso la superficie, come mostrato in Figure 1.10. Le seconda, chiamata *double-loops* si basa invece sulla creazione di bolle lunga una traiettoria circolare.
In particolare, nel WOA viene modellato matematicamente il primo dei due comportamenti, introducendo i seguenti tre meccanismi: accerchiamento della preda, bubble-net attack ed infine ricerca della preda [28].

1.6.1 Accerchiamento della preda

Dal momento che la soluzione ottima nello spazio di ricerca non è conosciuta, il WOA assume che l'attuale migliore soluzione, punto in cui è collocato un cetaceo, sia la preda target. A questo punto, gli altri individui aggiorneranno la loro posizione nei pressi dell'ottimo individuato. Tale comportamento viene riassunto dall'equazione 3.16 e dall'equazione 1.30

$$\mathbf{D} = |C\mathbf{X}^*(t) - \mathbf{X}(t)| \tag{1.29}$$

$$\mathbf{X}(t+1) = \mathbf{X}^*(t) - A\mathbf{D} \tag{1.30}$$

dove t indica la generica iterazione, A e C rappresentano dei parametri, $\mathbf{X}^*$ la posizione dell'ottimo, $\mathbf{X}$ il generico vettore posizione dell'individuo e $\mathbf{D}$ il vettore contenente il valore assoluto di ogni componente ottenuta attraverso l'Equazione 3.16. Il vettore $\mathbf{X}^*$ viene poi aggiornato ad ogni iterazione, qualora si trovasse una posizione migliore.
I parametri A e C vengono calcolati come riportato di seguito

$$A = 2ar_1 - a \tag{1.31}$$

$$C = 2r_2 \tag{1.32}$$

dove il parametro a è linearmente decrescente da due a zero durante le iterazioni ed i coefficienti r_1 e r_2 sono valori randomici nell'intervallo [0,1]. Considerando il caso di un probelma 1D, lo scalare r_1 può assumere come valore massimo 1 e come valore minimo 0: per cui il valore di A apparterrà all'intervallo [-a,a].

1.6.2 Bubble-net attack

Questa fase coincide con la fase di exploitation, ovvero di rifinitura della soluzione ottimale trovata. E' possibile definire due differenti approcci

1. Nel *meccanismo di restringimento avvolgente* la nuova posizione dell'agente di ricerca è definita in un qualsiasi punto presente tra la propria posizione e quella del miglior individuo

$$\mathbf{X}(t+1) = \mathbf{X}^*(t) - A\mathbf{D} \tag{1.33}$$

2. Nell'*aggiornamento a spirale* della posizione viene definita un' equazione che simula una traiettoria a spirale

$$\mathbf{X}(t+1) = \mathbf{D}'e^{bl}cos(2\pi l) + \mathbf{X}^*(t) \tag{1.34}$$

dove b è una costante per definire la forma della spirale logaritmica, l è un numero casuale all'interno dell'intervallo [-1,1] e D' è definito come

$$\mathbf{D}' = |\mathbf{X}^*(t) - \mathbf{X}(t)|$$

Come visto precedentemente, le megattere possono cacciare con due metodologie differenti. Per modellare tale comportamento si assume una probabilità del 50 % per scegliere fra le due strategie di caccia. Ovvero

$$\begin{cases} \mathbf{X}^* - A\mathbf{D} & \text{se } \mathrm{p} < 0.5 \\ \mathbf{D}'e^{bl}cos(2\pi l) + \mathbf{X}^*(t) & \text{se } \mathrm{p} \geq 0.5 \end{cases} \tag{1.35}$$

dove p è un numero casuale all'interno dell'intervalo [0,1].

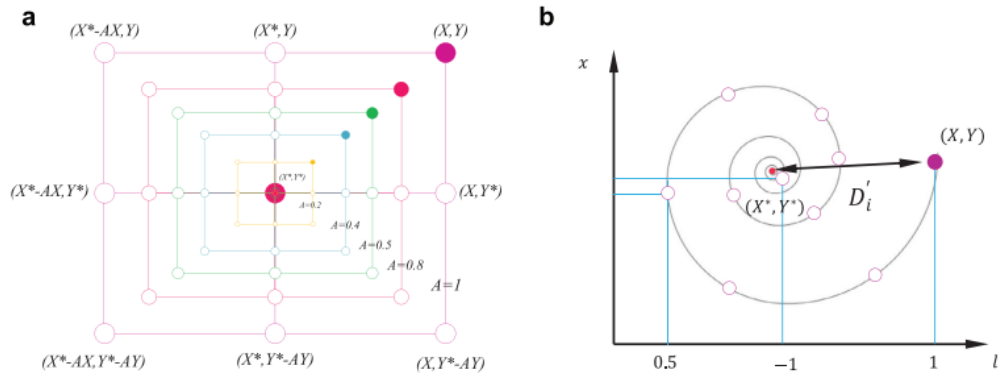

Figura 1.11: Approcci del bubble-net attack [28]

1.6.3 Ricerca della preda

La ricerca di una preda rappresenta la fase di exploration. Gli individui cercano in maniera randomica in accordo alla loro posizione reciproca.
Se A presenta valori maggiori di 1 o minori di -1, ovvero $|A| > 1$, viene forzato l'agente di ricerca a muoversi lontano dalla balena di riferimento. Rispetto alla fase di exploitation, la posizione degli individui viene aggiornata scegliendo una megattera di riferimento a caso anziché l'individuo al quale corrisponde la soluzione ottima.
Il modello matematico che governa tale fase è il seguente

$$\mathbf{D} = |C\mathbf{X}_{\mathbf{rand}} - \mathbf{X}| \tag{1.36}$$

$$\mathbf{X}(t+1) = \mathbf{X}_{\mathbf{rand}} - A\mathbf{D} \tag{1.37}$$

dove, appunto, $\mathbf{X}_{\mathbf{rand}}$ è il vettore che si riferisce ad un individuo a caso all'interno della popolazione.

1.6.4 Algoritmo WOA

Il WOA, rispetto agli altri algoritmi metaeuristici, presenta una importante capacità di esplorazione dovuto al modo in cui la posizione degli individui viene aggiornata. Infatti, l'Equazione 1.37, forza gli agenti di ricerca a muoversi in maniera casuale fra di loro mentre l'Equazione 1.33 e 1.34 permettono l'aggiornamento della posizione in maniera rapida, nella direzione della miglior soluzione individuata. Dal momento che queste due fasi vengono svolte in maniera indipendente il WOA riesce contemporaneamente sia ad evitare punti di minimo locale sia ad avere una buona velocità di convergenza all'ottimo. Invece, gli altri algoritmi metaeuristici utilizzano soltanto una metodologia per compiere entrambe le fasi di ricerca avendo

dunque una maggiore probabilità di incappare in un minimo locale.
Ovviamente il processo di randomizzazione del WOA determina un incremento del costo computazionale. Inoltre, la convergenza e la velocità con la quale le soluzioni vengono individuate dipendono dal parametro a che dunque ha un impatto elevato sulle performance dell'algoritmo.

Algorithm 2 Algoritmo WOA

▷ Creare ed inizializzare una popolazione di megattere di dimensione n_x
▷ Calcolare la fitness per ogni individuo
repeat
    for *ogni megattera i=1,. . ., n_s* **do**
        ▷ Aggiornare i seguenti parametri: a, A, C, l e p
        if p<0.5 **then**
            if |A| < 1 **then**
                ▷ Aggiornare la posizione degli individui con l'Equazione 1.33
            else
                ▷ Selezionare un individuo a caso X_{rand}
                ▷ Aggiornare la posizione degli individui con l'Equazione 1.37
            end if
        else
            ▷ Aggiornare la posizione dell'individuo con l'Equazione 1.34
        end if
    end for
    ▷ Riportare nel dominio gli individui che oltrepassano i limiti del dominio
    ▷ Calcolare la fitness di ogni individuo
    ▷ Aggiornare la posizione X^*
until *condizione di convergenza soddifsfatta*

1.6.5 Binary WOA

Come già visto nel Paragrafo 1.5.5 è possibile applicare gli algoritmi di ottimizzazione metaeuristici anche per risolvere problemi di natura discreta.
Nella versione continua dell'algoritmo, l'aggiornamento della posizione avviene mediante l'utilizzo dell'Equazione 1.33 o dell'Equazione 1.34 mentre nel BWOA questo implica il cambio di bit della stringa che si riferisce al generico individuo. Per cui, anche in questo caso è necessario definire una probabilità di cambiamento di un bit, come nel BPSO ed introdurre nuovamente le funzioni di trasferimento della Tabella 1.2.
Noto l'output delle funzioni di trasferimento, il cambio di bit avviene nel seguente

modo per le funzioni s-shape

$$\begin{cases} 0 & \text{se rand}() < \mathcal{S}(x_{id}) \\ 1 & \text{altrimenti} \end{cases} \tag{1.38}$$

e nel seguente modo per le v-shape

$$\begin{cases} (x_{id})^{-1} & \text{se rand}() < \mathcal{V}(x_{id}) \\ x_{id} & \text{altrimenti} \end{cases} \tag{1.39}$$

dove $(x_{id})^{-1}$ è il complemento di x_{id}.

Algorithm 3 Algoritmo BWOA

▷ Creare ed inizializzare una popolazione di megattere di dimensione n_x
▷ Calcolare la fitness per ogni individuo
repeat
    for *ogni megattera i=1,. . ., n_s* **do**
        ▷ Aggiornare i seguenti parametri: a, A, C, l, p
        if p<0.5 **then**
            if |A| < 1 **then**
                ▷ Aggiornare la posizione degli individui con l'Equazione 1.33
            else
                ▷ Selezionare un individuo a caso X_{rand}
                ▷ Aggiornare la posizione degli individui con l'Equazione 1.37
            end if
        else
            ▷ Aggiornare la posizione dell'individuo con l'Equazione 1.34
        end if
    end for
    ▷ Riportare nel dominio gli individui che oltrepassano i limiti del dominio
    ▷ Calcolare la fitness di ogni individuo
    ▷ Aggiornare la posizione X^*
until *condizione di convergenza soddifsfatta*

Capitolo 2

Shape sensing

Per Structural Health Monitoring (SHM) si intende il processo di implementazione di strategie per la monitorazione, tipicamente in real-time, del degrado del materiale e del cambio di geometrie di una struttura al fine di valutarne l'integrità durante le condizioni operative. I dati ottenuti possono dunque apportare una maggior sicurezza e maggiore efficienza nelle operazioni di manutenzione.
Aspetto ancor più importante è l'utilizzo di tale metodologia per strutture in materiale composito il quale progetto, effettuato con approccio damage-tolerance, risulta fortemente influenzato dalla tipologia del danno tipicamente diverso rispetto a quello di un qualsiasi altro materiale isotropo. Ad esempio, uno dei principali difetti, dipendenti dal processo di produzione, è la mancata adesione tra fibra e matrice o tra i vari strati del laminato così come la presenza di porosità. Per cui, avere la possibilità di poter monitorare in real-time la struttura apporterebbe un significativo miglioramento nelle performance e nella vita operativa di quest'ultima. Quello che si potrebbe effettuare è quindi una ricostruzione dei campi di temperatura, deformazione o spostamento della struttura attraverso l'utilizzo di una rete di sensori collocati in opportuni punti. L'output fornito da quest'ultimi fornirebbe inoltre un feed-back utile per l'implementazione dei sistemi di attuazione e controllo delle cosiddette *morphed wings* ovvero ali geometricamente adattive che riescono a cambiare forma in maniera autonoma durante il volo al fine di ridurre la resistenza aerodinamica o allargare l'inviluppo di volo del velivolo.
In particolare, lo shape sensing ovvero la ricostruzione del campo di spostamenti a partire dalla misura delle deformazioni rappresenta matematicamente un problema inverso che per definizione non soddisfa necessariamente le condizioni di esistenza, univocità e stabilità. Riguardo l'univocità, potrebbero esistere infinite configurazioni di sensori capaci di determinare il medesimo campo di spostamento. Circa la stabilità, piccoli disturbi nelle misure, tipicamente delle deformazioni, potebbero determinare grandi cambiamenti nella soluzione.

Differenti sono le metodologie che permettono la ricostruzione del campo di spostamenti a partire da misure sperimentali. Foss e Haugse in [29], a partire dalla misura sperimentale di alcuni parametri modali determinano il campo di spostamenti di una trave in alluminio a sbalzo.
Ko et al. [30] basandosi sulle misure sperimentali delle deformazioni assiali e relativo calcolo delle curvature, determina attraverso una doppia integrazione la deformata di una struttura alare.
Alexander Tessler in [31] fornisce una formulazione variazionale basata sulla minimizzazione dei minimi quadrati sfruttando un set completo di misure di deformazione di una piastra analizzata mediante una teoria FSDT includendo dunque le deformazioni membranali, flessionali ed a taglio trasversale.
In [32] viene definito un approccio variazionale allo shape sensing in cui un funzionale, basato sui minimi quadrati è discretizzato usando gli elementi finiti. Quest'ultimo rappresenta l'errore ai minimi quadrati tra le deformazioni misurate sperimentalmente e quelle ricavate mediante FEM. Minimizzando il funzionale si ottiene un sistema algebrico le cui incognite rappresentano i gradi di libertà del modello della struttura considerata. Ottenute le incognite nodali ed usando delle opportune funzioni di forma è poi possibile ottenere il campo di spostamento nel dominio.

2.1 Metodo modale

Introdotto in [29], il metodo usa le forme modali come base per esprimere il campo di spostamenti.
A partire dalla discretizzazione agli elementi finiti di una struttura è possibile esprimere in funzione delle $\mathcal{M}$ coordinate modali $\mathbf{q}$ il campo di spostamenti come

$$\mathbf{w} = [\Phi_d]\mathbf{q} \tag{2.1}$$

e quello delle deformazioni come

$$\boldsymbol{\epsilon} = [\Phi_s]\mathbf{q} \tag{2.2}$$

dove $\mathbf{w}$ è il vettore dei gradi di libertà di dimensione (D×1), $\boldsymbol{\epsilon}$ il vettore delle deformazioni di dimensione (S×1), $[\Phi_d]$ la matrice modale di dimensione (D×M) e $[\Phi_s]$ la matrice di dimensione (S×M) in cui la i-esima colonna rappresenta l'i-esimo set di deformazioni inerente alla i-esima forma modale del modello agli elementi finiti.
Invertendo l'Equazione 2.2

$$\mathbf{q} = [\Phi_s]^{-1}\boldsymbol{\epsilon} \tag{2.3}$$

è possibile ottenere il vettore dei gradi di libertà come

$$\mathbf{w} = [\Phi_d][\Phi_s]^{-1}\boldsymbol{\epsilon} \tag{2.4}$$

L'inversione della matrice $[\Phi_s]$ è però possibile se essa è quadrata, ovvero $S = \mathcal{M}$. Nelle applicazioni sperimentali è però insolito che il numero di sensori disponibili S sia uguale a quello dei modi calcolati $\mathcal{M}$. Viene dunque definita la relativa matrice pseudo-inversa. In particolare, data una generica matrice $[A]$ di dimensione ($N\times$M), una matrice $[A]^+$ di dimensione ($M\times$N) è definita pseudo-inversa di $[A]$ se

$$[A]^+ = [A]'[A]^{-1}[A]' \tag{2.5}$$

qualora il rango di [A] fosse massimo e N≥M.
Nelle situazioni pratiche, tipicamente il numero di sensori è maggiore del numero delle forme modali individuate per cui il problema risulta ben posto in quanto è presente un numero di equazioni maggiore del numero di incognite. In tal caso è possibile esprimere il vettore dei gradi di libertà come

$$\mathbf{w} = [\Phi_d]\left([\Phi_s]'\Phi_s]\right)^{-1}[\Phi_s]'\boldsymbol{\epsilon} \tag{2.6}$$

Chiaramente, il numero dei modi scelto deve essere sufficiente a descrivere in maniera adeguata il campo di spostamenti della struttura una volta applicatoci il carico.

2.2 Teoria di Ko

Basata sulla EBBT la teoria di Ko [30] permette di valutare la deformata di una trave a partire dalla misura sperimentale delle deformazioni assiali.

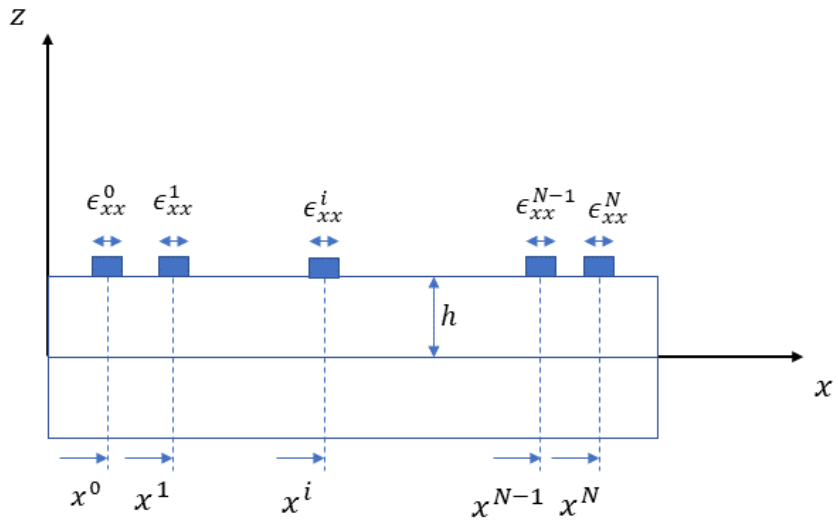

Figura 2.1: Geometria e posizione dei sensori

Sia la deformazione assiale ϵ_{xx}

$$\epsilon_{xx}(x) = -z w_{,xx}(x) \tag{2.7}$$

dove z è la distanza tra l'asse neutro della trave ed il sensore.
Assumendo la deformazione assiale lineare tra due punti di misura x_i

$$\epsilon_{xx}(x) = \epsilon_{xx}^{i-1}(x) + \frac{(\epsilon_{xx}^{i}(x) - \epsilon_{xx}^{i-1}(x))}{(x^i - x^{i-1})}\left(x - x^{i-1}\right) \tag{2.8}$$

con $i = 1, \ldots, N$ e $x^{i-1} \leq x \leq x^i$.
L'equazione 2.8 è sostituita nell'Equazione 2.7 e procedendo con la doppia integrazione nella variabile x si ottiene la seguente relazione per descrivere la deformata della trave

$$\begin{aligned} w_i(x) = -\frac{1}{6h}\Bigg[& \sum_{j=1}^{i}\left(2\epsilon_{xx}^{j-1}(x) + \epsilon_{xx}^{j}(x)\right)\left(x^j - x^{j-1}\right)^2 + \\ & +3\sum_{k=1}^{k-1}\left(\epsilon_{xx}^{k-1}(x) + \epsilon_{xx}^{k}(x)\right)\left(x^k - x^{k-1}\right)\left(x^i - x^k\right) \end{aligned} \tag{2.9}$$

La precedente relazione è ottenuta imponendo un incastro per $x = 0$.

2.3 Metodo agli elementi finiti inversi 1D

La risoluzione del problema inverso permette la ricostruzione del campo di spostamenti tridimensionale della trave a partire dalle misure sperimentali delle deformazioni.

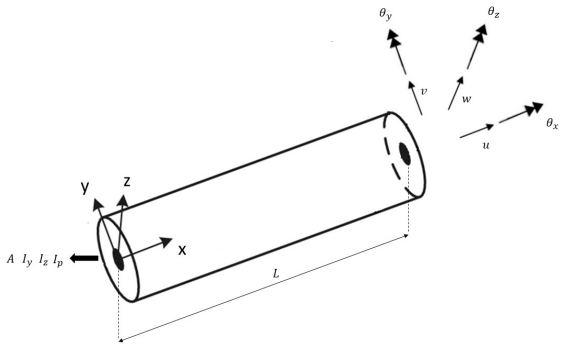

Figura 2.2: Notazione per la trave

Si supponga l'asse elastico coincidente con l'asse x perpendicolare alla sezione

definita nel piano {y,z} dove y e z sono i relativi assi principali d'inerzia. La trave ha inoltre lunghezza L e sezione A, caratterizzata dai momenti di inerzia I_y, I_z e dal momento di inerzia polare $I_p = I_y + I_z$. Si consideri infine un materiale omogeneo isotropo caratterizzato da un modulo di Young E, da un modulo di taglio G e da un modulo di Poisson ν.
In accordo alla teoria cinematica di Timoshenko, il campo di spostamenti 3D può essere espresso come

$$\begin{aligned} u_x(x,y,z) &= u(x,y,z) + z\theta_y(x,y,z) - y\theta_z(x)(x,y,z) \\ u_y(x,y,z) &= v(x,y,z) - z\theta_x(x,y,z) \\ u_z(x,y,z) &= w(x,y) + y\theta_x(x,y) \end{aligned} \tag{2.10}$$

dove u_x, u_y e u_Z sono gli spostamenti lungo gli assi x, y e z con u, v e w che indicano gli spostamenti valutati per y=z=0 ed infine θ_x, θ_y e θ_z le rotazioni attorno gli assi x, y e z.
Le sei variabili cinematiche possono dunque essere raggruppate nel vettore $\mathbf{u}$

$$\mathbf{u} = \{u, v, w, \theta_x, \theta_y, \theta_z\}' \tag{2.11}$$

Sotto l'ipotesi di piccoli deformazioni è possibile legare le deformazioni assiale ed a taglio trasversale alle sei deformazioni di sezione e_i

$$\epsilon_x(x) = e_1(x) + ze_2(x) + ye_3(x) \tag{2.12}$$

$$\gamma_{xz}(x) = e_4(x) + ye_6(x) \tag{2.13}$$

$$\gamma_{xy}(x) = e_5(x) - z_6(x) \tag{2.14}$$

Quest'ultime possono poi essere scritte in forma vettoriale come segue

$$\mathbf{e}(\mathbf{u}) = \{e_1, e_2, e_3, e_4, e_5, e_6\}' \tag{2.15}$$

$$\mathbf{e}(\mathbf{u}) = \{u_{,x}(x), \theta_{y,x}(x), -\theta_{z,x}(x), w_{,x}(x) + \theta_y(x), v_{,x}(x) - \theta_z(x), \theta_{x,x}(x)\}' \tag{2.16}$$

Viene ora introdotto il funzionale basato su un principio variazionale ai minimi quadrati [33]

$$\phi(\mathbf{u}) = ||\mathbf{e}(\mathbf{u}) - \mathbf{e}^\epsilon||^2 \tag{2.17}$$

dove con $\mathbf{e}^\epsilon$ si indica il vettore contenente le deformazioni sperimentali.
Il funzionale può essere espresso, per ogni elemento finito, attraverso la seguente sommatoria

$$\phi(\mathbf{u}) = \frac{L_e}{N} \sum_{i=1}^{N} [e_k(x_i) - (e_k^\epsilon)_i]^2 \tag{2.18}$$

con $k = 1, \ldots, n_s$. In particolare, N rappresenta il numero totale degli elementi, L_e la lunghezza dell'elemento trave, x_i le posizioni in cui vengono misurate le deformazioni e n_s il numero totale delle deformazioni valutate lungo la sezione.
Il funzionale di errore totale dell'elemento $\phi^{\mathbf{e}}$ è poi valutato prendendo in considerazione il vettore dei pesi dimensionali usato per imporre una correlazione più o meno forte tra le deformazioni $\mathbf{e}(\mathbf{u})$ e quelle sperimentali $\mathbf{e}^{\epsilon}$.

$$\phi^e = \mathbf{w}\phi = \sum_{k=1}^{n_s} w_k^e \phi_k^e \tag{2.19}$$

con

$$\mathbf{w}_k = \{w_1^0, w_2^0(I_y^e/A^e), w_3^0(I_z^e/A^e), w_4^0, w_5^0, w_6^0(I_p^e/A^e)\} \tag{2.20}$$

con $k = 1, \ldots, 6$.
Questo funzionale può essere usato in una discretizzazione agli elementi finiti in cui il campo di spostamenti è interpolato attraverso funzioni di forma continue appartenenti alla classe $\mathcal{C}^0$

$$\mathbf{u}(x) \approx \mathbf{u}^h = \mathbf{N}(x)\mathbf{u}^e \tag{2.21}$$

dove $[\mathbf{N}(x)]$ indica la matrice delle funzioni di forma, $\mathbf{u}^e$ i gradi di libertà nodali del generico elemento e x la coordinata assiale dell'elemento.
Sotto le ipotesi di piccole deformazioni, le deformazioni analitiche possono essere ottenute mediante le seguenti relazioni geoemtriche

$$\mathbf{e}(\mathbf{u}) = \mathbf{B}(x)\mathbf{u}^e \tag{2.22}$$

dove $\mathbf{B}$(x) è la matrice delle funzioni di forma derivate.
Sostituendo l'Equazione 2.22 nell'Equazione 2.21 e minimizzando il funzionale è possibile ottenere il seguente set di equazioni algebriche

$$\mathbf{k}^e\mathbf{u}^e = \mathbf{f}^e \tag{2.23}$$

dove i gradi di libertà nodali $\mathbf{u}^e$ rappresentano l'incognita del problema. La matrice $\mathbf{k}^e$ ed il vettore $\mathbf{f}^e$ possono essere espressi in funzione delle varie componenti di strain come

$$\mathbf{k}^e = \sum_{k=1}^{n_s} w_k^e \mathbf{k}_k^e \tag{2.24}$$

$$\mathbf{f}^e = \sum_{k=1}^{n_s} w_k^e \mathbf{f}_k^e \tag{2.25}$$

La matrice $\mathbf{k}^e$ è solo funzione della posizione dei sensori all'interno del generico elemento e $\mathbf{f}^e$ è sia funzione della posizione dei sensori sia del valore sperimentale delle misure di deformazione.

Tali quantità possono anche essere definite in funzione delle matrici delle funzioni di forma derivate

$$\mathbf{k}^e = \frac{L_e}{N}\sum_{i=1}^{N}[\mathbf{B}'_k(x_i)\mathbf{B}_k(x_i)] \tag{2.26}$$

$$\mathbf{f}^e = \frac{L_e}{N}\sum_{i=1}^{N}[\mathbf{B}'_k(x_i)(e^{\epsilon}_k)_i)] \tag{2.27}$$

Applicando il cambiamento di coordinate ed assemblando tutti gli elementi, è possibile ottenere il sistema algebrico generale

$$\mathbf{KU} = \mathbf{F} \tag{2.28}$$

La matrice $\mathbf{K}$ risulta non singolare, se applicate in maniera corretta le condizioni al contorno, in modo da impedire ogni moto di corpo rigido.

2.3.1 Elemento iTM2D0

L'interpolazione dell'elemento è sviluppata da Gherlone et al. in [33]. A partire dalle equazioni di equilibrio è possibile stabilire che lungo l'elemento le deformazioni e_1, e_4, e_5, e_6 sono costanti l'elemento mentre le deformazioni e_2 e e_3 risultano essere lineari.

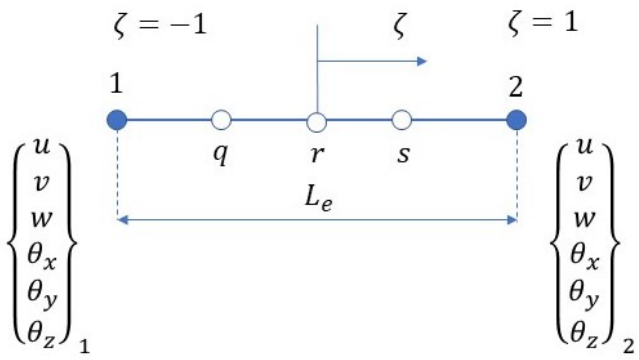

Figura 2.3: Elemento iTM2d0

Per cui, dalle relazioni geometriche è possibile osservare come u e θ_x siano lineari, θ_y e θ_z paraboliche, ed infine v e w cubiche.
Queste quantità possono dunque essere espresse in funzione dei seguenti polinomi interpolanti

$$u(\zeta) = \sum_{i=1,2} L^{(1)}(\zeta)u_i \qquad v(\zeta) = \sum_{i=1,2} L^{(1)}(\zeta)v_i - \sum_{j=1,r,2} N^{(3)}(\zeta)_j\theta_{zj} \tag{2.29}$$

$$w(\zeta) = \sum_{i=1,2} L^{(1)}(\zeta) w_i - \sum_{j=1,r,2} N^{(3)}(\zeta)_j \theta_{yj} \qquad \theta_x(\zeta) = \sum_{i=1,2} L^{(1)}(\zeta)_i \theta_{xi} \tag{2.30}$$

$$\theta_y(\zeta) = \sum_{j=1,r,2} L^{(2)}(\zeta)_j \theta_{yj} \qquad \theta_z(\zeta) = \sum_{j=1,r,2} L^{(2)}(\zeta)_j \theta_{zj} \tag{2.31}$$

dove $L^{(1)}(\zeta)_i$ $(i = 1{,}2)$ e $L^{(2)}(\zeta)_j$ $(i = 1, r{,}2)$ sono, rispettivamente, i polinomi lineari e quadratici di Lagrange mentre $N^{(3)}(\zeta)_j$ $(i = 1{,}2)$ sono polinomi cubici. I dettagli dei precedenti polinomi sono riportati in Appendice A.

2.4 Metodo agli elementi finiti inversi 2D

La risoluzione del problema inverso permette la ricostruzione del campo di spostamenti tridimensionale della piastra a partire dalle misure sperimentali delle deformazioni sulle superfici superiore ed inferiore del corpo opportunamente vincolate ed a prescindere dalla tipologia di materiale e dalla distribuzione del carico. Si consideri una piastra di forma qualunque rappresentata in Figura 2.4 definita nel seguente dominio

$$\Omega = \{(x, y, z) \subset \mathcal{R}^3 : \quad z \in [-t, t],\ (x, y) \in \mathcal{A} \subset \mathcal{R}^2\}$$

dove $2t$ è lo spessore della piastra, $\mathcal{A}$ l'area del piano medio posizionato in $z = 0$

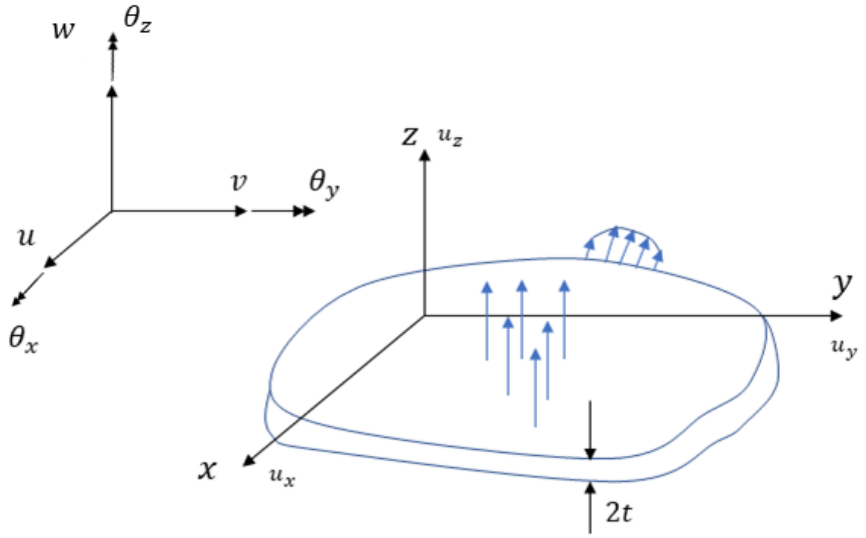

Figura 2.4: Notazione per la piastra

ed s il contorno. Il corpo è inoltre soggetto sia a carichi nel piano che fuori dal piano.

Le deformazioni vengono misurate sulla parte superiore ed inferiore del corpo ovvero nei punti aventi coordinate

$$\mathbf{x_i} = (x_i, y_i, \pm t)$$

dove $i = 1, \ldots, N$. In ogni punto è possibile quindi definire le tre componenti di deformazione nel piano

$$\epsilon = (\epsilon_{xx}^{\pm}, \epsilon_{yy}^{\pm}, \gamma_{xy}^{\pm})$$

dove l'apice " $+$ " si riferisce alle deformazioni valutate sulla faccia superiore della piastra mentre l'apice " $-$ " a quelle valutate sulla faccia inferiore.
In ogni punto della piastra è possibile definire lo spostamento attraverso un vettore costituito da tre componenti. In accordo alla teoria FSDT, è possibile definire ognuna di quest'ultime come

$$\begin{aligned} u_x(x, y, z) &= u(x, y) + z\theta_y(x, y) \\ u_y(x, y, z) &= v(x, y) - z\theta_x(x, y) \\ u_z(x, y, z) &= w(x, y) \end{aligned} \tag{2.32}$$

dove $u(x, y)$, $v(x, y)$ e $w(x, y)$ rappresentano gli spostamenti dei punti appartenenti al piano medio nelle direzioni x, y e z; $\theta_x(x, y)$ e $\theta_y(x, y)$ le rotazioni lungo l'asse x e lungo l'asse y.
Mediante l'utilizzo degli operatori differenziali $[\mathcal{L}_\epsilon]$, $[\mathcal{L}_k]$ e $[\mathcal{L}_\gamma]$ definiti come

$$[\mathcal{L}_\epsilon] = \begin{bmatrix} \dfrac{\partial}{\partial x} & 0 & 0 & 0 & 0 \\ 0 & \dfrac{\partial}{\partial y} & 0 & 0 & 0 \\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} & 0 & 0 & 0 \end{bmatrix} \qquad [\mathcal{L}_k] = \begin{bmatrix} 0 & 0 & 0 & 0 & \dfrac{\partial}{\partial x} \\ 0 & 0 & 0 & -\dfrac{\partial}{\partial y} & 0 \\ 0 & 0 & 0 & -\dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} \end{bmatrix}$$

$$[\mathcal{L}_\gamma] = \begin{bmatrix} 0 & 0 & \dfrac{\partial}{\partial x} & 0 & 1 \\ 0 & 0 & \dfrac{\partial}{\partial y} & -1 & 0 \end{bmatrix}$$

è possibile ottenere l'espressione delle relazioni cinematiche che legano le deformazioni al campo di spostamento

$$\begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} = [\mathcal{L}_\epsilon] \begin{Bmatrix} u \\ v \\ w \\ \theta_x \\ \theta_y \end{Bmatrix} + [\mathcal{L}_k] \begin{Bmatrix} u \\ v \\ w \\ \theta_x \\ \theta_y \end{Bmatrix} = \begin{Bmatrix} \epsilon_{x0} \\ \epsilon_{y0} \\ \gamma_{xy0} \end{Bmatrix} + z \begin{Bmatrix} k_{x0} \\ k_{y0} \\ k_{xy0} \end{Bmatrix} = \mathbf{e}(\mathbf{u}) + z\mathbf{k}(\mathbf{u}) \tag{2.33}$$

Le deformazioni a taglio trasversale vengono invece definite come

$$\mathbf{g}(\mathbf{u}) = \begin{Bmatrix} \gamma_{xz0} \\ \gamma_{y0z} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & \dfrac{\partial}{\partial x} & 0 & 1 \\ 0 & 0 & \dfrac{\partial}{\partial y} & 1 & 0 \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \\ \theta_x \\ \theta_y \end{Bmatrix} = [\mathbf{L}_\gamma]\mathbf{u} \tag{2.34}$$

Quindi, è possibile definire il campo di deformazioni come

$$\mathbf{e} = \{u_{,x}, v_{,x}, v_{,x} + u_{,y}\}' = \{\epsilon_1, \epsilon_2, \epsilon_3\}' \tag{2.35}$$

$$\mathbf{k} = \{\theta_{y,x}, -\theta_{x,y}, \theta_{y,y} - \theta_{x,c}\}' = \{\epsilon_4, \epsilon_5, \epsilon_6\}' \tag{2.36}$$

$$\mathbf{w} = \{w_{,x} + \theta_y, w_{,y} - \theta_x\}' = \{\epsilon_7, \epsilon_8\}' \tag{2.37}$$

Con la discretizzazione del problema agli elementi finiti inversi, per ogni elemento inverso, e, il vettore dei gradi di libertà (DOF) $\mathbf{u}^e$ è definito come

$$\mathbf{u}^e = \{\mathbf{u}_1^e, \mathbf{u}_2^e, \ldots, \mathbf{u}_n^e\}' \tag{2.38}$$

dove $\mathbf{u}_i^e$ rappresenta il vettore dei DOF di ogni nodo dell'elemento e. In particolare, adottando come teoria della piastra una FSDT, a ciascun nodo è riferibile il seguente set di gradi di libertà

$$\mathbf{u}_i^e = \{u_i, v_i, w_i, \theta_{xi}, \theta_{yi}\}$$

Introducendo la matrice delle funzioni di forma $[\mathbf{N}]$ è possibile interpolare il campo di spostamento all'interno del generico elemento inverso

$$\{u, v, w, \theta_x, \theta_y\} = [\mathbf{N}]\mathbf{u}^e \tag{2.39}$$

Inserendo l'Equazione 2.39 nelle Equazioni 2.35, 2.36 2.37 è possibile esprimere le deformazioni in funzione dei gradi di libertà e delle derivate delle funzioni di forma

$$\epsilon_k(\mathbf{u}^e) = [\mathbf{B}_k]\mathbf{u}^e \tag{2.40}$$

dove $k = 1, \ldots, 8$ e $[\mathbf{B}_k]$ è la matrice che contiene le derivate delle funzioni di forma inerenti alla k-esima deformazione misurata.
In particolare è poi possibile definire un elemento inverso a tre nodi ed uno a 4 nodi, discussi nei paragrafi 2.4.1 e 2.4.2. Si definisce *iMIN3* l'elemento inverso triangolare a tre nodi a deformazione costante in cui per ogni nodo è possibile definire i seguenti DOF

$$\mathbf{u}_i^e = \{u_i, v_i, w_i, \theta_{xi}, \theta_{yi}\} \tag{2.41}$$

L'elemento a quattro nodi, definito *iQS4* presenta invece sei DOF per nodo

$$\mathbf{u}_i^e = \{u_i, v_i, w_i, \theta_{xi}, \theta_{yi}, \theta_{zi}\} \tag{2.42}$$

dove il grado di libertà θ_{zi} che rappresenta la rotazione attorno l'asse z è un *drilling* DOF.
In particolare, applicando l'Equazione 2.33 nei punti $(x_i, y_i, \pm t)$, dove le deformazioni vengono misurate, è possibile esprimere le deformazioni membranali e le curvature del piano medio come

$$\mathbf{e}_i^\epsilon = \begin{Bmatrix} \epsilon_{x0}^\epsilon \\ \epsilon_{y0}^\epsilon \\ \epsilon_{xy0}^\epsilon \end{Bmatrix}_i = \frac{1}{2}\left(\begin{Bmatrix} \epsilon_{xx}^+ \\ \epsilon_{yy}^+ \\ \gamma_{xy}^+ \end{Bmatrix}_i + \begin{Bmatrix} \epsilon_{xx}^- \\ \epsilon_{yy}^- \\ \gamma_{xy}^- \end{Bmatrix}_i \right) \tag{2.43}$$

$$\mathbf{k}_i^\epsilon = \begin{Bmatrix} k_{x0}^\epsilon \\ k_{y0}^\epsilon \\ k_{xy0}^\epsilon \end{Bmatrix}_i = \frac{1}{2t}\left(\begin{Bmatrix} \epsilon_{xx}^+ \\ \epsilon_{yy}^+ \\ \gamma_{xy}^+ \end{Bmatrix}_i - \begin{Bmatrix} \epsilon_{xx}^- \\ \epsilon_{yy}^- \\ \gamma_{xy}^- \end{Bmatrix}_i \right) \tag{2.44}$$

con $i = 1, \ldots, N$ dove N indica il numero totale dei punti in cui effettuare le misure, $\{\epsilon_{xx}^+, \epsilon_{yy}^+, \epsilon_{xx}^+\}'$ e $\{\epsilon_{xx}^-, \epsilon_{yy}^-, \epsilon_{xx}^-\}'$ le deformazioni nel piano misurate sulla faccia superiore ed inferiore della piastra.

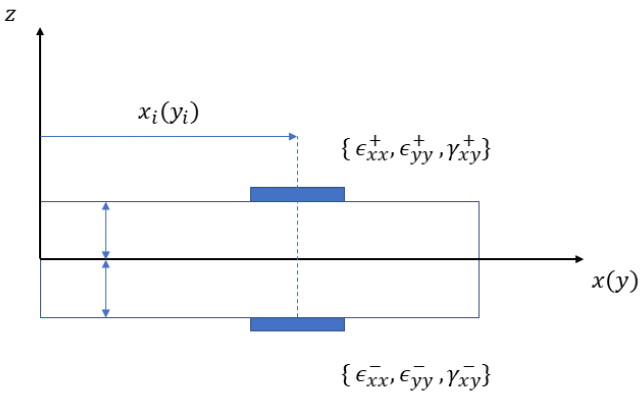

Figura 2.5: Rappresentazione grafica delle deformazioni misurate

Inoltre, se vale la simmetria del materiale rispetto al piano medio è possibile prevedere le seguenti modalità di deformazione

1. Sotto l'azione di un carico trasversale, la piastra è solamente sottoposta a flessione con una distribuzione a farfalla delle deformazioni rispetto al piano

medio dove $\mathbf{e}=\mathbf{0}$. Dato che le deformazioni variano solo per il segno è possibile effettuare la misura o sulla faccia superiore o su quella inferiore della piastra.

2. Sotto l'azione di carichi nel piano, la componente flessionale risulta nulla $\mathbf{k}=\mathbf{0}$. Anche in questo caso è possibile effettuare la misura su una delle due facce.

3. Nel caso in cui sono applicate varie tipologie di carico sono necessarie le misure su entrambe le facce.

Le misure sperimentali possono essere poi interpolate su tutta la superficie del piano medio attraverso tecniche di curve-fitting e curve-smoothing dove con smoothing intendiamo il processo di rimozione del rumore dai dati appena acquisiti sperimentalmente.
Un possibile approccio basato sugli elementi finiti è lo Smoothing Element Analysis (SEA) che permette di ottenere un campo di deformazioni rappresentato attraverso una funzione polinomiale definita a tratti contenuta nello spazio $\mathcal{C}^1$ avente derivata del primo ordine appartenente alla classe $\mathcal{C}^0$. Questa classe di funzioni permette di ricavare le derivate del primo ordine delle curvature $\mathbf{k}^\epsilon$ per poi ottenere le deformazioni a taglio trasversale $\mathbf{g}^\epsilon$.
Viene ora introdotto il funzionale basato su un principio variazionale dei minimi quadrati, utilizzato per la ricostruzione del campo di spostamenti tridimensionale a partire da misure sperimentali delle deformazioni

$$\Psi^e(\mathbf{u}^e) = \sum_{k=1}^{8} \lambda_k^e w_k^e \iint_{A_e} \left(\epsilon_k(\mathbf{u}^e) - \epsilon_k^\epsilon)\right)^2 \, dxdy \tag{2.45}$$

Nell'integrale doppio è possibile osservare la differenza, elevata al quadrato, delle deformazioni calcolate analiticamente $\epsilon_k(\mathbf{u}^e)$ e di quelle sperimentali $\epsilon_k^\epsilon(\mathbf{u}^e)$, definite nelle Equazioni 2.43 e 2.43. Viene introdotto un fattore di penalizzazione λ_k^e, tipicamente dell'ordine del 10^{-4} o 10^{-6} che ha il compito di porre a zero, o comunque ad un valore sufficientemente piccolo, la misurazione sperimentale della deformazione in un particolare elemento non coperto da una possibile distribuzione di sensori. Di contro, è posto chiaramente ad 1 se la misura nell'elemento viene effettuata. Il coefficiente w_k^e serve invece a garantire la consistenza dimensionale dell'integrando. Nello specifico, $w_k^e = 1$ per k=1, 2, 3, 7, 8 e $w_k^e = (2t)^2$ per k=4, 5, 6 dove $2t$ è lo spessore dell'elemento. Ciò dipende dal fatto che le curvature $\mathbf{k}$ hanno dimensione [1/m].
L'integrale doppio è calcolato numericamente attraverso la quadratura di Gauss: scelto il numero di punti di Gauss n il calcolo dell'integrale viene espresso attraverso la seguente sommatoria

$$\iint_{A_e} \left(\epsilon_k(\mathbf{u}^e) - \epsilon_k^\epsilon)\right)^2 \, dxdy = \sum_{g=1}^{n\times n} J(g)\omega_g \left(\epsilon_k(\mathbf{u}^e) - \epsilon_k^\epsilon)\right)^2 \tag{2.46}$$

per $k = 1, \ldots, 8$. Per eseguire il calcolo è indispensabile conoscere, oltre che i pesi di quadratura ω_g funzioni del numero dei punti di Gauss, il determinante dello Jacobiano della trasformazione dalle coordinate fisiche a quelle naturali dell'elemento calcolato in ogni punto di quadratura g.
Minimizzando il funzionale è possibile ottenere un set di equazioni lineari algebriche

$$\frac{\partial \Phi_e(\mathbf{u}^e)}{\partial \mathbf{u}^e} = [\mathbf{k}^e]\mathbf{u}^e - \mathbf{f}^e = 0$$

$$[\mathbf{k}^e]\mathbf{u}^e = \mathbf{f}^e \tag{2.47}$$

dove la matrice $[\mathbf{k}^e]$ è funzione della posizione dei sensori mentre il vettore $\mathbf{f}^e$ è una funzione delle misure ottenute. Come è possibile osservare, è presente un'analogia con la matrice di rigidezza ed il vettore delle forze dell'analisi agli elementi finiti. In particolare, sia la matrice $[\mathbf{k}^e]$ che il vettore $\mathbf{f}^e$ vengono poi calcolate nel seguente modo

$$[\mathbf{k}^e] = \sum_{k=1}^{8} \sum_{g=1}^{n \times n} [J(g)\lambda_k^e w_k^e \omega_g [\mathbf{B}_{k(g)}]' [\mathbf{B}_{k(g)}]] \tag{2.48}$$

$$\mathbf{f}^e = \sum_{k=1}^{6} \sum_{g=1}^{n \times n} [J(g)\lambda_k^e w_k^e \omega_g [\mathbf{B}_{k(g)}]' \epsilon_{k(g)}^{\epsilon}] \tag{2.49}$$

dove il pedice g si riferisce al calcolo della matrice $[\mathbf{B}_k]$ ed alla misura sperimentale ϵ^{ϵ} nei punti di Gauss. Assemblando i singoli contributi di ogni elemento ed effettuando il cambio di coordinate in quelle globali attraverso la matrice di trasformazione $[\mathbf{T}^e]$ è possibile ottenere la formulazione algebrica globale del problema

$$[\mathbf{K}]\mathbf{U} = \mathbf{F} \tag{2.50}$$

dove

$$[\mathbf{K}] = \sum_{i=1}^{N_e} [\mathbf{T}^e]' [\mathbf{k}^e] [\mathbf{T}^e] \tag{2.51}$$

$$\mathbf{F} = \sum_{i=1}^{N_e} [\mathbf{T}^e]' \mathbf{f}^e \tag{2.52}$$

$$\mathbf{U} = \sum_{i=1}^{N_e} [\mathbf{T}^e]' \mathbf{u}^e \tag{2.53}$$

con N_e che indica il numero totale di elementi inversi usati per discretizzare il dominio.
Applicando le opportune condizioni al contorno per evitare moti di corpo rigido della struttura la matrice $[\mathbf{K}]$ risulta invertibile e quindi è possibile determinare in maniera univoca il vettore dei gradi di libertà $\mathbf{U}$

$$\mathbf{U} = [\mathbf{K}]^{-1}\mathbf{F} \tag{2.54}$$

A determinare la non singolarità della matrice [**K**], oltre che i vincoli, contribuiscono anche sia il numero dei sensori utilizzati che la loro distribuzione sulla mesh del modello.

2.4.1 Elemento iMIN3

L'elemento iMIN3 [34] si basa sulla teoria $FSDT$ ed il campo di spostamento all'interno dell'elemento è espresso attraverso funzioni di forma di classe $\mathcal{C}^0$ che interpolano in maniera non isoparametrica i 5 DOF dei tre nodi che costituiscono l'elemento. In particolare, il grado di libertà w è interpolato da un polinomio del quarto ordine mentre le altre quattro incognite linearmente. La matrice delle funzioni di forma è quindi definibile come

$$[\mathbf{N}] = [[\mathbf{N}_1][\mathbf{N}_2][\mathbf{N}_3]] \tag{2.55}$$

dove la singola matrice $[\mathbf{N}_i]$ è definita come

$$[\mathbf{N}_i] = \begin{bmatrix} \mathcal{L}_i & 0 & 0 & 0 & 0 \\ 0 & \mathcal{L}_i & 0 & 0 & 0 \\ 0 & 0 & \mathcal{L}_i & \mathcal{L}_{2i} & \mathcal{L}_{1i} \\ 0 & 0 & 0 & \mathcal{L}_i & 0 \\ 0 & 0 & 0 & 0 & \mathcal{L}_i \end{bmatrix} \tag{2.56}$$

dove le funzioni $\mathcal{L}_i$ con $i = 1,2,3$ sono i polinomi lineari interpolanti in coordinate parametriche del triangolo mentre $\mathcal{L}_{1i}$ e $\mathcal{L}_{2i}$ con $i = 1,2,3$ sono le seguenti funzioni di forma quadratiche

$$\mathcal{L}_{1i} = \frac{\mathcal{L}_i}{2}(a_l\mathcal{L}_m - a_m\mathcal{L}_l) \tag{2.57}$$

$$\mathcal{L}_{2i} = \frac{\mathcal{L}_i}{2}(b_m\mathcal{L}_l - b_l\mathcal{L}_m) \tag{2.58}$$

$$a_k = (x_m - x_l), b_k = (y_l - y_m) \tag{2.59}$$

dove $x_{m,l}$ e $y_{m,l}$ sono le coordinate del (m,l)-esimo nodo del triangolo. I pedici sono ottenuti attraverso una permutazione ciclica di $k = 1,2,3$, $l = 2,3,1$ e $m = 3,1,2$.

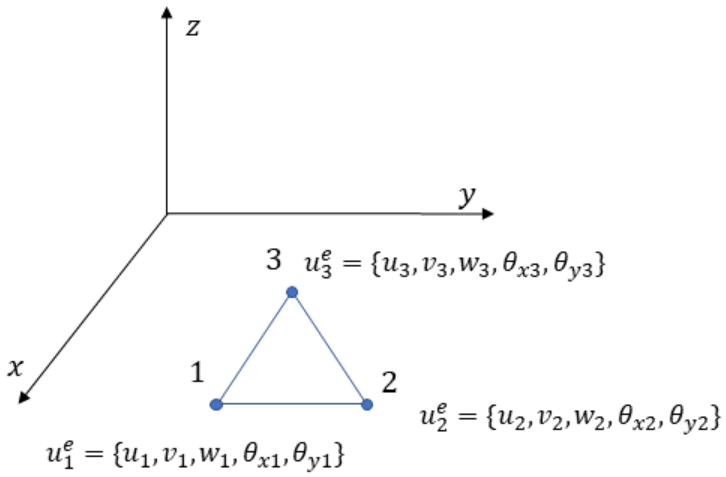

Figura 2.6: Elemento iMIN3

Effettuando le derivate delle funzioni di forma è possibile ottenere le seguenti matrici, utilizzate per calcolare il campo di deformazioni descritto dall' Equazioni 2.35, 2.36 e 2.37

$$[\mathbf{B}_k] = [[\mathbf{B}_k^1], [\mathbf{B}_k^2], [\mathbf{B}_k^3]] \tag{2.60}$$

dove il pedice k indicizza le otto deformazioni

$$[\mathbf{B}_1^i] = \{\mathcal{L}_{i,x}, 0, 0, 0, 0\} \tag{2.61}$$

$$[\mathbf{B}_2^i] = \{0, \mathcal{L}_{i,y}, 0, 0, 0\} \tag{2.62}$$

$$[\mathbf{B}_3^i] = \{\mathcal{L}_{i,y}, \mathcal{L}_{i,x}, 0, 0, 0\} \tag{2.63}$$

$$[\mathbf{B}_4^i] = \{0, 0, 0, 0, \mathcal{L}_{i,x}\} \tag{2.64}$$

$$[\mathbf{B}_5^i] = \{0, 0, 0, -\mathcal{L}_{i,y}, 0\} \tag{2.65}$$

$$[\mathbf{B}_6^i] = \{0, 0, 0, -\mathcal{L}_{i,x}, \mathcal{L}_{i,y}\} \tag{2.66}$$

$$[\mathbf{B}_7^i] = \{0, 0, \mathcal{L}_{i,x}, \mathcal{L}_{2i,x}, \mathcal{L}_{1i,x} + \mathcal{L}_i\} \tag{2.67}$$

$$[\mathbf{B}_8^i] = \{0, 0, \mathcal{L}_{i,y}, \mathcal{L}_{2i,i} - \mathcal{L}_i, \mathcal{L}_{1i,y}\} \tag{2.68}$$

Per un elemento triangolare lo Jacobiano è costante e pari all'area dell'elemento A^e. Inoltre, essendo le funzioni $\mathcal{L}_i$ lineari e le uniche coinvolte nel calcolo delle deformazioni per l'incapacità sperimentale di misurare le deformazioni a taglio

trasversale, le deformazioni all'interno del singolo dominio risultano essere costanti. Per cui, la matrice $[\mathbf{k}^e]$ ed il vettore $\mathbf{f}^e$ vengono calcolate come

$$[\mathbf{k}^e] = \sum_{k=1}^{6} A^e[\lambda_k^e w_k^e [\mathbf{B}_k]'[\mathbf{B}_k]] + \sum_{k=8}^{8} A^e \sum_{g=1}^{n\times n} [\lambda_k^e w_k^e \omega_g [\mathbf{B}_{k(g)}]'[\mathbf{B}_{k(g)}]] \tag{2.69}$$

$$\mathbf{f}^e = \sum_{k=1}^{6} A^e[\lambda_k^e w_k^e [\mathbf{B}_k] \epsilon_{k(centroid)}^{\epsilon}] \tag{2.70}$$

Il pedice *centroid* si riferisce al fatto che la misura sperimentale è effettuata nel centroide dell'elemento: si evince sperimentalmente che così facendo le misurazioni risultano essere più accurate. In particolare, dal momento che le deformazioni sono costanti all'interno di un elemento iMIN3 allora è possibile effettuare una sola misura per la valutazione delle deformazioni.

2.4.2 Elemento iQS4

Così come l'elemento iMIN3 anche l'elemento iQS4 [34] è basato sulla FSDT ed usa una formulazione non isoparametrica per l'interpolazione del campo di spostamenti attraverso l'uso di funzioni di forma lineari ($\mathbf{N}_i$) e paraboliche ($\mathcal{L}_i, \mathcal{M}_i$).
La matrice delle funzioni di forma è quindi definibile come

$$[\mathbf{N}] = [[\mathbf{N}_1][\mathbf{N}_2][\mathbf{N}_3][\mathbf{N}_4]] \tag{2.71}$$

dove la singola matrice $[\mathbf{N}_i]$ è definita come

$$[\mathbf{N}_i] = \begin{bmatrix} \mathcal{N}_i & 0 & 0 & 0 & 0 & \mathcal{L}_i \\ 0 & \mathcal{N}_i & 0 & 0 & 0 & \mathcal{M}_i \\ 0 & 0 & \mathcal{N}_i & -\mathcal{L}_i & -\mathcal{M}_i & 0 \\ 0 & 0 & 0 & \mathcal{N}_i & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathcal{N}_i & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathcal{N}_i \end{bmatrix} \tag{2.72}$$

con i = 1,2,3,4. Le funzioni di forma sono riportate in Appendice B.
Effettuando le derivate delle funzioni di forma è possibile ottenere le seguenti matrici, utilizzate per calcolare il campo di deformazioni descritto dall' Equazioni 2.35, 2.36 e 2.37

$$[\mathbf{B}_k] = [[\mathbf{B}_k^1], [\mathbf{B}_k^2], [\mathbf{B}_k^3], [\mathbf{B}_k^4]] \tag{2.73}$$

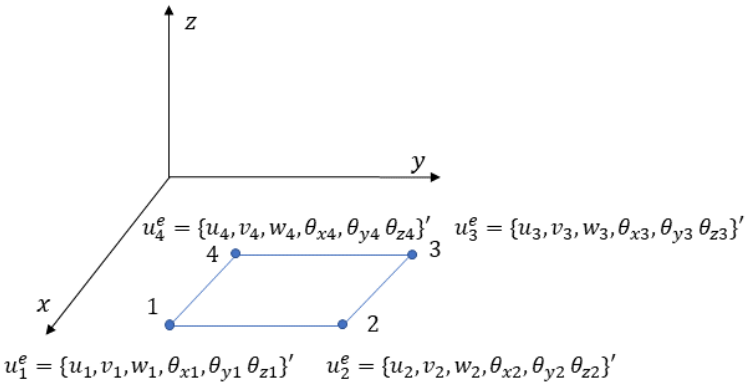

Figura 2.7: Elemento iQS4

dove il pedice k indicizza le otto deformazioni

$$[\mathbf{B}_1^i] = \{\mathcal{N}_{i,x}, 0, 0, 0, 0, \mathcal{L}_{i,x}\} \tag{2.74}$$

$$[\mathbf{B}_2^i] = \{0, \mathcal{N}_{i,y}, 0, 0, 0, \mathcal{M}_{i,y}\} \tag{2.75}$$

$$[\mathbf{B}_3^i] = \{\mathcal{N}_{i,y}, \mathcal{N}_{i,x}, 0, 0, 0, (\mathcal{L}_{i,y} + \mathcal{M}_{i,x})\} \tag{2.76}$$

$$[\mathbf{B}_4^i] = \{0, 0, 0, 0, \mathcal{N}_{i,x}, 0\} \tag{2.77}$$

$$[\mathbf{B}_5^i] = \{0, 0, 0, -\mathcal{N}_{i,y}, 0, 0\} \tag{2.78}$$

$$[\mathbf{B}_6^i] = \{0, 0, 0, -\mathcal{N}_{i,x}, -\mathcal{N}_{i,y}, 0\} \tag{2.79}$$

$$[\mathbf{B}_7^i] = \{0, 0, \mathcal{N}_{i,x}, -\mathcal{L}_{i,x}, (-\mathcal{M}_{1i,x} + \mathcal{N}_i), 0\} \tag{2.80}$$

$$[\mathbf{B}_8^i] = \{0, 0, \mathcal{N}_{i,y}, (-\mathcal{L}_{i,y} - \mathcal{N}_i), \mathcal{M}_{i,y}, 0\} \tag{2.81}$$

A differenza dell'elemento triangolare, l'elemento iQS4 non presenta deformazioni costanti nel proprio dominio, per cui l'espressione della formula della quadratura di Gauss non può essere semplificata. Inoltre, per evitare un numero eccessivo di sensori, la misura della deformazione in un qualunque punto all'interno dell'elemento viene associata a tutti i punti di Gauss. Con la seguente semplificazione, si ottengono le seguenti formule

$$[\mathbf{k}^e] = \sum_{k=8}^{8} \sum_{g=1}^{n \times n} [J(g) \lambda_k^e w_k^e \omega_g [\mathbf{B}_{k(g)}]' [\mathbf{B}_{k(g)}]] \tag{2.82}$$

$$\mathbf{f}^e = \sum_{k=1}^{6} \sum_{g=1}^{n \times n} [J(g) \lambda_k^e w_k^e [\mathbf{B}_k] \epsilon_{k(element)}^e] \tag{2.83}$$

dove con il pedice *element* si indica una posizione arbitraria all'interno del dominio. Per incrementare l'accuratezza è possibile utilizzare un numero dispari di punti di Gauss: così facendo, uno di questi è posizionato sul centroide dell'elemento. Si introduce un fattore di penalizzazione χ_g posto ad 1 per il punto posizionato sul centroide in cui la misura della deformazione è molto accurata mentre assume valori bassi per i rimanenti punti di quadratura. Le formule vengono quindi modificate nel seguente modo

$$[\mathbf{k}^e] = \sum_{k=8}^{8} \sum_{g=1}^{n\times n} [J(g)\lambda_k^e w_k^e \omega_g \chi_g [\mathbf{B}_{k(g)}]' [\mathbf{B}_{k(g)}]] \tag{2.84}$$

$$\mathbf{f}^e = \sum_{k=1}^{6} \sum_{g=1}^{n\times n} [J(g)\lambda_k^e w_k^e \chi_g [\mathbf{B}_k] \epsilon^{\epsilon}_{k(element)}] \tag{2.85}$$

Chiaramente i pesi χ_g possono solo essere utilizzati solo se nota la posizione esatta dell'estensimetro. Di conseguenza, tale approccio non può essere utilizzato per la deformazioni a taglio trasversale che non possono essere misurate sperimentalmente.

Capitolo 3

Ottimizzazione della disposizione dei sensori mediante PSO e WOA

Sperimentalmente si evince che la distribuzione ed il numero dei sensori impiegati influenzano l'accuratezza con la quale il campo di spostamenti, a partire dalla misura delle deformazioni, viene determinato.
In [35] Esposito e Gherlone affrontano il problema utilizzando un algoritmo genetico per la ricerca di una configurazione ottimale di sensori da disporre su di un cassone alare in composito. In particolare, il codice lavora in due fasi distinte: nella prima viene adottato un criterio di selezione basato sulla fitness di ogni individuo mentre nella seconda ci si concentra sulla valutazione del ranking individuale. In questo modo, viene favorito inizialmente il processo di exploration per poi dare adito alla fase di exploitation.
Come funzione obiettivo è stata considerata la root mean square tra lo spostamento lungo la direzione z del modello FEM di riferimento e quello ricavato attraverso l'iFEM

$$\%ERMS_w = 100 \times \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{w_i - w_i^{ref}}{w_{max}^{ref}}\right)^2} \tag{3.1}$$

dove w_{max}^{ref} rappresenta lo spostamento massimo individuato ed i indicizza i nodi del modello numerico.
Nel seguente capitolo verrà discusso il modo in cui il WOA ed il PSO sono stati adattati al problema in esame. In particolare, per ogni metaeuristica è stata implementata sia una versione continua, lavorando su numeri reali, sia una discreta, dove è stata utilizzata una codifica in binario per la rappresentazione degli individui. Come si vedrà, tale scelta è stata dettata, oltre che dalla necessità di rendere più

complesso il problema, sia in termini di ERMS da minimizzare sia per il numero di possibili combinazioni, anche dalla prematura convergenza delle versione continue, sinonimo di limitata exploration.

Figura 3.1: Interfaccia tra funzione obiettivo e framework di ottimizzazione

Altro aspetto importante riguarda l'utilizzo dei vincoli ed il rispetto dei limiti del dominio di ricerca, ingredienti fondamentali di un'ottimizzazione. Ogni configurazione di sensori deve infatti rispettare un certo numero prefissato di componenti e l'algoritmo deve essere capace di fornire soluzioni *feasible*, ovvero accettabili, situate all'interno del campo di ricerca.

3.1 Spazio di ricerca

3.1.1 Ottimizzazione continua

Ad ogni nodo della mesh dell'iFEM è attribuito un numero di identificazione (ID), del quale sono note le coordinate rispetto ad un determinato sistema di riferimento e le relative deformazioni, che serviranno alla ricostruzione del campo di spostamenti. Nota dunque la posizione è possibile collocare sulla struttura reale i sensori, siano essi semplici rosette o delle fibre ottiche: nello specifico, i punti di interesse sono quelli dei centroidi degli elementi presenti nel modello numerico. Ad esempio, siano gli ID riportati in Tabella 3.1 ai quali corrisponde l'effettivo collocamento sulla struttura di Figura 3.2.
In ambiente MATLAB è stata dunque definita una cell array: in questo modo, è stato possibile operare contemporaneamente sia sul singolo ID, per le rosette, sia su di un loro insieme, per le fibre. Dunque, la ricerca è effettuata all'interno di uno spazio del quale gli individui sono costituiti da un raggruppamento di indici della suddetta struttura.

Tipologia di sensore	ID di riferimento
Rosetta	{169}
Rosetta	{193}
Rosetta	{601}
Fibra	{ 289 1 3:2:47 337:360 625}
Fibra	{265:288 325 326 601:624 661 }

Tabella 3.1: Esempio di ID per rosette e fibre

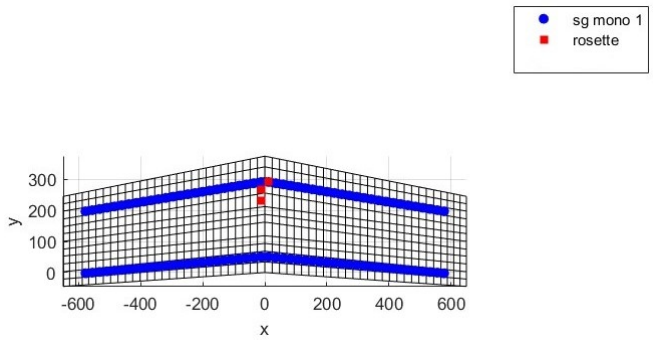

Figura 3.2: Rappresentazione grafica di una possibile configurazione di sensori

Nella versione continua del codice la cell array è costituita da 39 celle: 15 inerenti alle fibre e 24 alle rosette. Quest'ultime sono disposte nella parte centrale del pannello in quanto le deformazioni valutate in questa zona permettono di migliorare l'accuratezza del calcolo eseguito mediante iFEM. Inoltre, l'utilizzo delle fibre permette di per se un'ampia copertura e quindi un elevato valore delle misure di deformazione.

3.1.2 Ottimizzazione discreta

Anche in questo caso è stata utilizzata una cell array, all'interno della quale sono stati inseriti gli ID inerenti ai nodi dei centroidi. La principale differenza risiede nel fatto che ogni individuo è rappresentato dalle codifiche in binario degli indici della cell array. In particolare, codificando a 9 bit è stato necessario limitare il numero dei possibili candidati a 511, riuscendo comunque a garantire una quasi totale copertura del pannello. Tale scelta non influenza la capacità dell'algoritmo di individuare soluzioni migliori in quanto sono stati esclusi i nodi soggetti a lievi deformazioni.

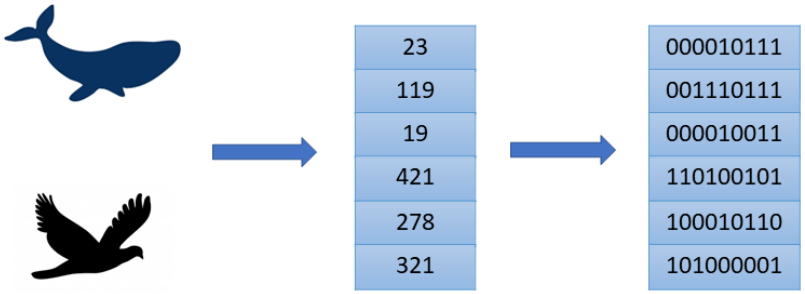

Figura 3.3: Esempio di codifica a 9 bit con individui costituiti da sei sensori

3.2 Estensione del dominio

È essenziale garantire che le nuove soluzioni generate rientrino all'interno dello spazio di ricerca. In letteratura [36], ciò è eseguito mediante tre differenti metodologie: random ((a) in Fig 3.4), d'assorbimento ((b) in Fig 3.4) ed a specchio ((c) in Fig 3.4).
Nel primo, l'individuo che viola i limiti dello spazio di ricerca viene sostituito con un individuo a caso. Nel secondo, viene sostituito con il limite, destro o sinistro, del dominio ed infine, nello schema a specchio, l'individuo viene riportato all'interno del campo di ricerca in maniera simmetrica rispetto il limite, destro o sinistro, di quest'ultimo.

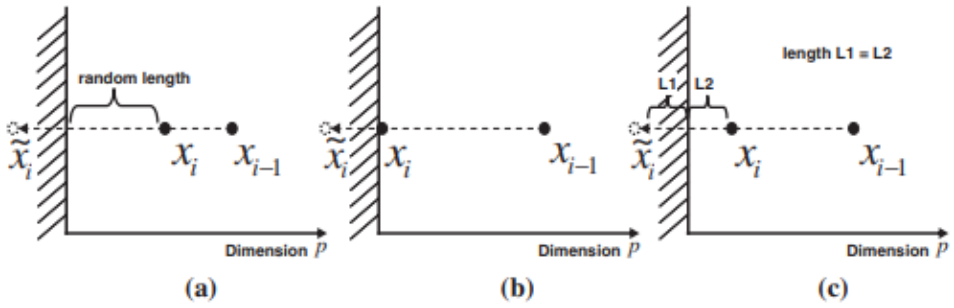

Figura 3.4: Spiegazione delle tre metodologie di handling boundary constraints [36]

Invece, la strategia utilizzata per entrambi gli algoritmi implementati, prevede l'utilizzo di una rimappatura del dominio ed è stata utilizzata esclusivamente per le versioni continue dei codici. In quelle binarie, la ricerca è effettuata andando a

variare i bit degli individui: una volta definita la codifica, è possibile esprimere solo i numeri che vanno da 1 a $2^{n_{bit}} - 1$, per cui risulta impossibile individuare soluzioni non accettabili.
In particolare, la rimappatura prevede un'estensione del dominio sufficiente a coprire i possibili valori numerici, positivi e negativi, che possono essere generati dall'algoritmo. Ricordando che lo spazio di ricerca è rappresentato dai valori numerici che indicano gli indici della cell array, sono state create due matrici di dimensione 49×39 che contengono al loro interno i valori numerici successivi a 39 ed inferiori ad 1.

40	41	42	...	78
79	80	81	...	117
...	...	...	...	...
1912	1913	1914	...	1950

Tabella 3.2: Matrice di estensione: limite superiore

0	-1	-2	...	-38
-39	-40	-81	...	-77
...	...	...	...	...
-1872	-1873	-1874	...	-1910

Tabella 3.3: Matrice di estensione: limite inferiore

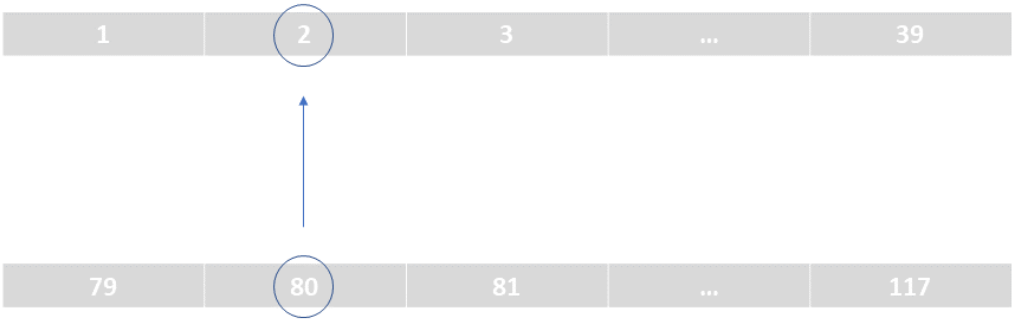

Figura 3.5: Esempio di rimappatura

Nel momento in cui viene generato un valore numerico incluso in una delle due matrici ne si considera l'indice della colonna relativa e lo si converte nel relativo

valore numerico posizionato nella medesima colonna del vettore {1,..., 39}.
Ad esempio, come osservabile in Figura 3.5, il numero 80 è il valore numerico presente nella posizione (2,2) della matrice di estensione del limite superiore e viene sostituito con il valore 2 anch'esso posizionato in seconda posizione dello spazio di ricerca.

3.3 Ottimizzazione vincolata

L'ottimizzazione vincolata introduce una qualsiasi condizione che limita il modo di essere o di svolgersi di un'azione. In particolare, un vincolo è espresso da una condizione che deve essere soddisfatta dalle soluzioni che si ricercano.
Uno dei metodi più utilizzati prevede l'utilizzo di una funzione di penalizzazione [37]

$$f(x) = \begin{cases} f(x) & \text{se } x \in \mathbf{D} \\ f(x) + f_{pen}(x) & altrimenti \end{cases} \tag{3.2}$$

dove $\mathbf{D}$ rappresenta l'insieme delle soluzioni accettabili e f_{pen} una particolare funzione da sommare a quella obiettivo qualora non dovessero essere rispettati i vincoli imposti.
Nel caso in cui debbano essere posizionate sia fibre che rosette sulla struttura, il processo di ricerca deve rispettare i seguenti vincoli

1. Un numero di rosette r ed un numero di fibre f
2. Nessuna ripetizione di sensori all'interno di una determinata configurazione

Per quanto riguarda il numero di sensori è stata considerata la distanza euclidea tra il numero effettivo f di fibre scelto dall'algoritmo e quello invece richiesto come input f^*. Definendo un parametro di penalizzazione $r \in [0,1]$, la funzione può dunque essere espressa come

$$f_{pen}(x) = f(x) + r\sqrt{(f - f^*)^2} \tag{3.3}$$

Nel momento in cui viene individuata una configurazione che prevede un numero non esatto di fibre e di rosette, viene sommato, alla valutazione della funzione obiettivo, un contributo tanto più grande quanto l'errato numero di sensori.
In questo modo la soluzione valutata presenterà una fitness molto elevata e difficilmente potrà essere considerata come una delle migliori individuate, lasciando il posto alle configurazioni nelle quali si riscontra un corretto numero di sensori.
Circa la presenza di sensori molteplici all'interno di una configurazione si è deciso di utilizzare semplicemente una sostituzione casuale in modo da poter garantire l'univocità di ogni sensore.

3.4 Arrotondamento

Sia il PSO che il WOA nascono per operare su funzioni obiettivo esplicite continue, per cui è stata necessaria l'introduzione di alcune modifiche al fine di poter generare soluzioni valide e corrette.
Dal momento che l'iFEM richiede l'ID dei centroidi, per poter sfruttarne le deformazioni, sperimentali o numeriche, è necessario correggere i valori numerici decimali che vengono generati dall'ottimizzazione. La scelta è ricaduta sull'utilizzo di un semplice arrotondamento, approssimando all'intero più vicino.
Quanto detto è in ogni caso da riferire solo alle versioni continue dei due codici in quanto la controparte binaria permette di mantenere valori interi a seguito della decodifica.
Per cui, se ad esempio viene generata la seguente configurazione:

$$\{123.23 \quad 328.87 \quad 217.78 \quad 121.13 \quad 98.93 \quad 12.12 \quad 79.43\} \tag{3.4}$$

a seguito dell'operazione di arrotondamento si ottiene

$$\{123 \quad 329 \quad 218 \quad 121 \quad 99 \quad 12 \quad 79\} \tag{3.5}$$

che rappresenta dunque una soluzione accettabile ai fini del calcolo tramite iFEM.

3.5 Implementazione del PSO

Prima di tutto, occorre definire i seguenti parametri di input:

1. Coefficiente della componente sociale c_2
2. Coefficiente della componente cognitiva c_1
3. Coefficiente d'inerzia w
4. Numero degli individui n
5. Limite superiore ub e inferiore dello spazio di ricerca lb
6. Fattore di penalizzazione r
7. Velocità massima V_{max}
8. Velocità minima V_{min}
9. Numero massimo di iterazioni $iter$

Inoltre, se la configurazione prevede l'utilizzo di differenti tipologie di sensori occorre definire sia il numero delle rosette che quello delle fibre da utilizzare.

3.5.1 Versione continua

Ciascun individuo è definito attraverso una *structure array* all'interno della quale sono stati inseriti i seguenti cinque differenti campi:

1. *Posizione*
2. *Velocità*
3. *Valutazione della fitness nella iterazione corrente*
4. *Miglior valutazione di sempre della fitness*
5. *Miglior posizione di sempre*

dove per posizione si intende la configurazione di sensori generata durante il susseguirsi delle iterazioni.
Prima operazione da svolgere è l'inizializzazione sia della velocità che della popolazione iniziale. La prima è effettuata impostando il valore di partenza a zero, la seconda attraverso un semplice campionamento casuale effettuato dallo spazio di ricerca, una volta definiti la tipologia ed il numero di sensori.

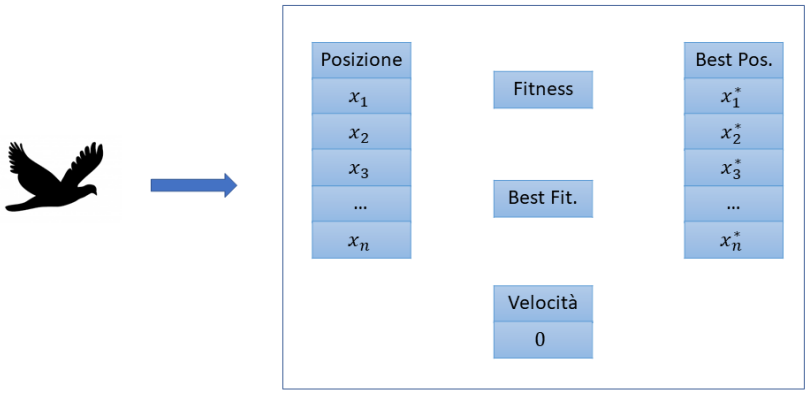

Figura 3.6: Individuo del PSO

Dopo di che, come visto nel Capitolo 1, va aggiornata la velocità di ogni individuo attraverso l'Equazione 1.17, che deve essere contenuta all'interno dell'intervallo $[V_{min}, V_{max}]$. In caso contrario, la velocità viene modificata come segue

$$V_i(t+1) = min(max(V_i(t), V_{min}), V_{max}) \tag{3.6}$$

Dopo di che si passa all'aggiornamento della posizione mediante l'Equazione 1.5 ed a seguito della valutazione della funzione obiettivo per ogni individuo, si definisce la particella migliore dell'iterazione, che verrà mantenuta finché una soluzione migliore non verrà individuata nelle iterazioni successive.

3.5.2 Versione binaria

Nel seguente paragrafo verrà affrontata l'analisi della versione binaria del Particle Swarm Optimization che, così come il BWOA, è stata utilizzata per ottimizzare la disposizione delle sole rosette sulla struttura.
In aggiunta ai parametri del Paragrafo 3.5.1 va definito in questo caso il numero di bit, indispensabile per la codifica degli individui. Inoltre sono state usate le seguenti leggi variabili per i parametri c_1, c_2 e w

$$c_1 = \frac{(c_{1,min} - c_{1,max})}{iter} t + c_{1,max} \tag{3.7}$$

$$c_2 = \frac{(c_{2,min} - c_{2,max})}{iter} t + c_{1,max} \tag{3.8}$$

$$w = (w_{in} - w_{fin}) \frac{iter - t}{iter} + w_{fin} \tag{3.9}$$

dove $c_{1,max}$=$c_{2,max}$=2, $c_{1,min}$=$c_{2,min}$=0.5, w_{in}=1.1 e w_{fin}=0.4.
La scelta di questi parametri, oltre che ad essere comunemente diffusi in letteratura, è dovuta al fatto di voler nelle fasi iniziali dare maggiore importanza all'exploration, per poi concentrarsi a migliorare la soluzione individuata quando si è prossimi al raggiungimento del numero massimo di iterazioni.
Anziché utilizzare una struct array, in questo caso è stato deciso di creare per ogni individuo due matrici ed un vettore: una inerente al posizionamento dei sensori, all'interno della quale sono state inserite le codifiche in binario degli indici della cell array del dominio, una per le velocità ed infine un vettore per salvare la valutazione della fitness di ogni individuo.
La logica di ricerca è la medesima della versione precedente del codice dunque basata sull'Equazione 1.17 per l'aggiornamento della velocità e sull'Equazione 1.5 per quanto riguarda l'aggiornamento della posizione. La differenza risiede nel fatto che le operazioni vengono eseguite su ogni bit che caratterizza la codifica della velocità di ogni particella.
A questo punto entra in gioco la funzione di trasferimento che riceve in input l'i-esima componente di velocità. L'output viene dunque comparato con un valore soglia campionato in maniera randomica dall'intervallo [0,1] al fine di decidere come modificare il bit della codifica della posizione del sensore.
In particolare, è stato deciso di implementare la funzione sigmoidea della famiglia s-shape

$$\mathcal{S}(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{3.10}$$

Sia ad esempio le seguente codifica a 9 bit di una generica posizione:

$$\mathbf{x}_i = \{010110011\} \tag{3.11}$$

Si consideri la terza componente della velocità, v_{i3}, pari a 0.345 ed il valore soglia, k, pari a 0.469. Seguendo la logica di trasferimento delle funzioni s-shape, riportata nel Paragrafo 1.5.5, essendo v_{i3} minore del valore k, il terzo bit della codifica viene cambiato in 1. Terminato l'aggiornamento, viene il tutto decodificato in modo da poter fornire l'input all'iFEM.

3.6 Implementazione del WOA

Prima di tutto, occorre definire i seguenti parametri di input:

1. Parametro per la forma della spirale b
2. Numero degli individui n
3. Limite superiore ub e inferiore dello spazio di ricerca lb
4. Fattore di penalizzazione r
5. Numero massimo di iterazioni $iter$

Anche in questo caso se la configurazione sensoristica è mista, è necessario definire il tipo ed il numero dei sensori.

3.6.1 Versione continua

Analogamente al PSO è necessario inizializzare una popolazione iniziale attraverso un campionamento randomico effettuato dallo spazio di ricerca. Ad ogni individuo viene fatta corrispondere una determinata configurazione di sensori.

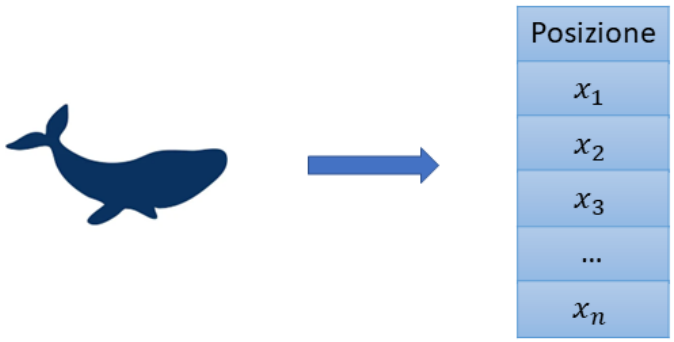

Figura 3.7: Individuo del WOA

Valutata la funzione obiettivo si prosegue aggiornando le posizioni mediante le

Equazioni 1.33, 1.34 e 1.37.
Come è possibile osservare, il framework rimane il medesimo del PSO: ciò che cambia sono esclusivamente le equazioni che vengono utilizzate per aggiornare le posizioni e quindi generare la nuova popolazione dell'iterazione successiva.

3.6.2 Versione binaria

Rispetto al BPSO, il BWOA presenta alcune modifiche. Implementando infatti una prima versione del codice, andando dunque a sfruttare la codifica delle possibili soluzioni, ciò che si è riscontrato è stata una prematura convergenza della soluzione ad un valore dell'ERMS piuttosto elevato.
Un primo tentativo è stato quello di testare le differenti funzioni di trasferimento appartenenti sia alla famiglia s-shape che alla v-shape ma senza alcun miglioramento. Si è deciso dunque di introdurre una variazione alla struttura originale dell'algoritmo al fine di poter garantire una maggiore eterogeneità degli individui, ispirandosi al lavoro di Nadimi-Shahraki et Al. [38] nello sviluppo del Binary Enhanced WOA (BE-WOA).
In particolare, viene creato un pool di k individui P_i=($P_{i,1}$, $P_{i,1}$, ..., $P_{i,d}$) attraverso l'Equazione 3.12 per generare un insieme di soluzioni mediante un meccanismo di crossover tra la peggiore soluzione e quella ottimale individuata.

$$\mathbf{P}_i^t = \mathbf{B}_i^t \times \mathbf{X}_{brnd}^t + \bar{\mathbf{B}}_i^t \times \mathbf{X}_{worst}^t \tag{3.12}$$

Vengono introdotti dei nuovi vettori rispetto alla versione originale del BWOA: i vettori binari $\mathbf{B}_i^t$ e $\bar{\mathbf{B}}_i^t$ dei quali l'ultimo rappresenta il complemento del primo, la peggior t-esima soluzione $\mathbf{X}_{worst}$ ed il vettore posizione $\mathbf{X}_{brnd}$ definito come

$$\mathbf{X}_{brnd}^t = rand \times (\delta_{best\_max} - \delta_{best\_min}) + \delta_{best\_max} \tag{3.13}$$

dove δ_{best_max} e δ_{best_min} sono i limiti superiore ed inferiori della soluzione $\mathbf{X}_{best}^t$.
Dopo aver inizializzato casualmente la matrice binaria $\mathbf{B}^t$=($\mathbf{B}_1^t, \mathbf{B}_1^t, \ldots, \mathbf{B}_N^t$), una percentuale della popolazione pari al 20% viene selezionata randomicamente e sottoposta al cosiddetto meccanismo di migrazione attuato attraverso la seguente equazione

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{rnd}^t - \mathbf{X}_{brnd}^t \tag{3.14}$$

$$\mathbf{X}_{rnd}^t = rand \times (\delta_{max} - \delta_{min}) + \delta_{min} \tag{3.15}$$

dove δ_{max} e δ_{min} sono i limiti superiore ed inferiore dello spazio di ricerca.
Definito un valore randomico ρ_i compreso all'interno dell'intervallo (0,1), la restante parte della popolazione viene aggiornata come segue

1. Se ρ_i<0.5 e $|\mathbf{A}_i| < 0.5$ viene utilizzata la strategia dell'accerchiamento della preda descritta nel Paragrafo 1.6.1 con la differenza che il parametro vettoriale

$\mathbf{D}^t$ viene ora definito in funzione di un individuo campionato randomicamente $\mathbf{P}^t_{rnd3}$

$$\mathbf{D} = |C\mathbf{X}^*(t) - \mathbf{P}^t_{rnd3}| \tag{3.16}$$

2. Se ρ_i<0.5 e $|\mathbf{A}_i| \geq 0.5$ viene utilizzata la cosiddetta strategia di ricerca preferenziale che permette di aumentare la capacità di explorating dell'algoritmo. Vengono definiti due individui random $\mathbf{P}^t_{rnd1}$ e $\mathbf{P}^t_{rnd2}$ e le posizioni aggiornate come segue

$$\mathbf{X}^{t+1}_i = \mathbf{X}^t_i + \mathbf{A}^t_i \times (\mathbf{C}^t_i \times \mathbf{P}^t_{rnd1} - \mathbf{P}^t_{rnd2}) \tag{3.17}$$

   In particolare, il parametro $\mathbf{A}^t_i$ è campionato da una distribuzione di Cauchy definita da un parametro di scala σ=0.1 e di posizione μ=0.5. La scelta di questa distribuzione è dettata dal fatto che i valori campionati risultano essere relativamente grandi, maggiori dunque rispetto a quelli di una semplice distribuzione normale. In questo modo viene migliorata la capacità dell'algoritmo di esplorare nuove zone dello spazio di ricerca.

3. Se $\rho_i \geq 0.5$ viene utilizzato il bubble-net attack descritto nel Paragrafo 1.6.2

Per quanto riguarda la funzione di trasferimento, viene ora introdotta la classe di funzioni $U - shape$ attraverso la seguente equazione

$$U(x^t_{i,j}) = \alpha \times |x^{beta}_{i,j}| \tag{3.18}$$

dove α e β regolano la pendenza e la larghezza del bacino della funzione. La scelta di questi parametri influenza le fasi di exploration ed exploitation.
Infine, il BE-WOA adotta la seguente legge per il cambio del bit

$$b^t_{i,j} = \begin{cases} 1 & \text{se } U(x^t_{i,j}) \leq rand(0,1) \\ 0 & \text{se } U(x^t_{i,j}) > rand(0,1) \end{cases} \tag{3.19}$$

Per quanto riguarda il codice di ottimizzazione implementato sono state apportate alcune modifiche

1. Modifica nella definizione dell'individuo $\mathbf{X}^t_{rnd}$

2. Modifica dell'inizializzazione del pool

In particolare, per poter inizializzare il pool di individui, costituito sempre dal 20 % della popolazione, vengono casualmente selezionati alcuni sensori della miglior configurazione individuata. Inoltre, lo stesso numero di sensori è randomico e campionato dall'intervallo [1,15]: per cui, possono essere selezionati al massimo 15 sensori o al minimo 1 sensore. I rimanenti sensori della configurazione vengono poi selezionati casualmente dallo spazio di ricerca.

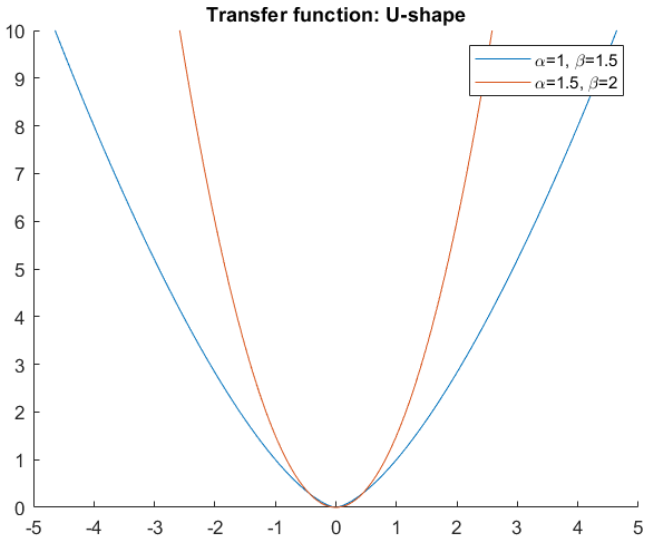

Figura 3.8: Funzione di trasferimento: U-shape

Dopodichè, definito casualmente l'individuo $\mathbf{X}_{rnd}^{t}$ vengono modificate le codifiche dei sensori nel seguente modo

$$\mathbf{X}_{i}^{t+1} = |\mathbf{X}_{rnd}^{t} - \mathbf{X}_{brnd}^{t}| \tag{3.20}$$

Il modulo serve soltanto a convertire i valori unitari negativi nei corrispettivi valori positivi. Inoltre, tale scelta è stata adottata anche in ogni equazione che presenta una differenza, intesa come operazione aritmetica, tra codifiche in binario.
Quest'operazione è eseguita esclusivamente per il 20 % degli individui, valore comunque modificabile, della popolazione mentre la restante parte deriva direttamente dal lavoro di ricerca svolto dall'algoritmo.

3.7 Comparazione tra GA, PSO e WOA

3.7.1 Ottimizzazione con fibre e rosette

I due algoritmi implementati sono stati utilizzati per l'ottimizzazione della disposizione di otto fibre e sei rosette ed i risultati comparati con quanto ottenuto da un algoritmo genetico. I run sono stati eseguiti settando il numero massimo di iterazioni ed il numero di individui in modo da avere un valore pressoché uguale delle valutazioni effettuate mediante iFEM, in particolare 7000. Se con il PSO ed il WOA si perviene a questo risultato andando semplicemente a moltiplicare il numero di individui e quello delle iterazioni impostate, per il GA è stato necessario

variare in maniera opportuna sia il numero degli individui che quello delle iterazioni: infatti, per ogni iterazione viene effettuata un'operazione di single-point crossover e two-point crossover tra gli individui andando dunque a duplicare le valutazioni effettuate attraverso l'iFEM ad ogni iterazione. In aggiunta, vanno incluse ulteriori valutazioni dovute alla generazione di nuove popolazioni per mutazione ed inversione che avvengono con una certa probabilità: per questo motivo non è possibile, come nel WOA e PSO, stabilire con esattezza il numero delle valutazioni effettuate ma è comunque possibile stimarne il totale.
Al fine di individuare il giusto setup in termini di input da fornire agli algoritmi è stato deciso di considerare sia il caso di 28 individui e 250 iterazioni che quello di 14 individui e 500 iterazioni.

	GA	PSO	WOA
Numero rosette	6	6	6
Numero fibre	8	8	8
Individui	50	14	14
Numero di iterazioni	68	500	500
Valutazioni iFEM	∼7000	7000	7000

Tabella 3.4: Primo setup per l'ottimizzazione della disposizione di fibre e rosette

	GA	PSO	WOA
Numero rosette	6	6	6
Numero fibre	8	8	8
Individui	100	28	28
Numero di iterazioni	33	250	250
Valutazioni iFEM	∼7000	7000	7000

Tabella 3.5: Secondo setup per l'ottimizzazione della disposizione di fibre e rosette

In Tabella 3.7 e Tabella 3.6 è possibile osservare le medie delle fitness ottimizzate e del numero di valutazioni dell'iFEM necessarie a raggiungerle.

	GA	PSO	WOA
Media fitness ottenuta [%]	0.2934	0.2192	0.1801
Media valutazioni iFEM	3780	3162	3456

Tabella 3.6: Medie della fitness e delle valutazioni iFEM: 14 individui

	GA	PSO	WOA
Media fitness ottenuta [%]	0.2271	0.2300	0.2498
Media valutazioni iFEM	3980	4740	3998

Tabella 3.7: Medie della fitness e delle valutazioni iFEM: 28 individui

In Figura 3.9 è raffigurato l'andamento della miglior fitness individuata in funzione del numero di iterazioni di una delle dieci analisi effettuate, riportate in Appendice C.

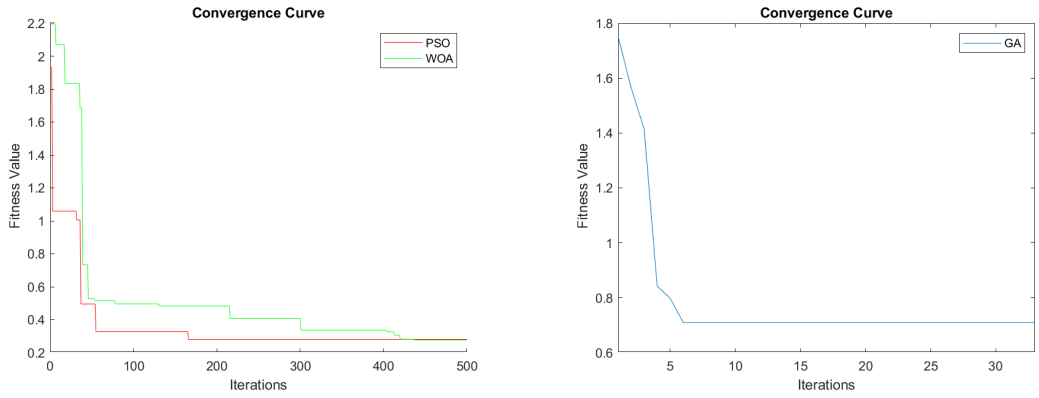

Figura 3.9: Comparazione ottimizzazione di fibre e rosette: 14 individui e 500 iterazioni

Come è possibile osservare dai dati riportati, per entrambi i setup i risultati sono fra di loro comparabili sia in termini della miglior fitness ottenuta che per quanto riguarda il numero di valutazioni necessarie per ottenerla. In particolare, il risultato migliore in termini di fitness, pari a 0.1801 %, si ottiene con il WOA considerando un numero inferiore di individui ed uno maggiore di iterazioni così come per quanto riguarda il numero di valutazioni necessarie, pari a 3162, dove è però il PSO il framework di ottimizzazione migliore.
Chiaramente le differenze sono minime parlando di variazioni che si hanno sul decimo o centesimo percentuale: tutti e tre gli algoritmi sembrano dunque comportarsi in maniera analoga.

3.7.2 Ottimizzazione con sole rosette

Analogamente al caso precedente, l'ottimizzazione è stata svolta settando il numero massimo di iterazioni ed il numero di individui in modo da poter avere un valore simile delle valutazioni effettuate attraverso il calcolo agli elementi finiti inversi, in

particolare 800000. Stesse considerazioni possono poi essere effettuate sul numero di valutazioni delle configurazioni ottenute mediante il GA.

	GA	PSO	WOA
Numero rosette	20	20	20
Individui	200	200	200
Numero di iterazioni	2000	4000	4000
Valutazioni iFEM	∼800000	800000	800000

Tabella 3.8: Setup per l'ottimizzazione delle rosette

A titolo di esempio viene riportata in Figura 3.10 una delle dieci ottimizzazioni effettuate in termini di andamento della miglior fitness individuata in funzione del numero di iterazioni, i quali risultati sono poi riportati in Appendice D.

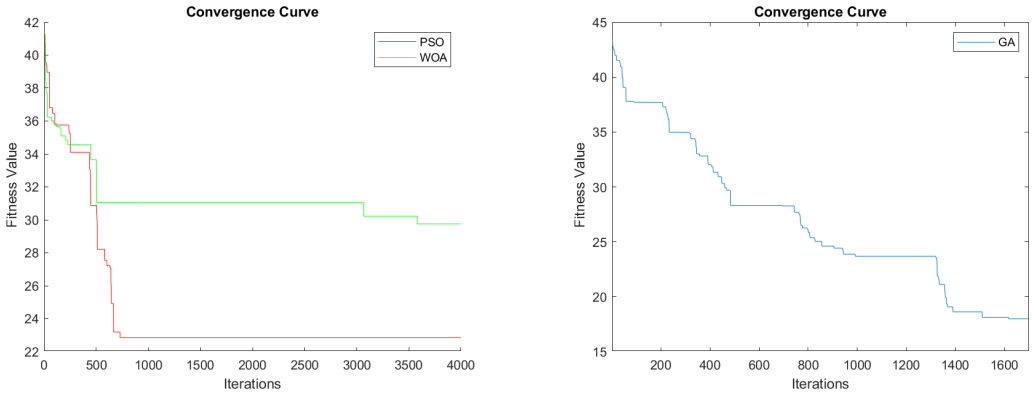

Figura 3.10: Comparazione ottimizzazione rosette

	GA	PSO	WOA
Media fitness ottenuta [%]	19.8323	27.2488	29.0437
Media valutazioni iFEM	582040	124080	589560

Tabella 3.9: Medie della fitness e delle valutazioni iFEM per la disposizione delle rosette

Ciò che si può subito osservare è che il GA permette di ottenere i migliori risultati in termini di fitness ottimizzata rispetto al WOA e PSO per i quali viene difficile oltrepassare un valore dell'errore di circa il 27 %. Di contro, tende a raggiungere

tali valori dopo un numero elevato di iterazioni, principalmente rispetto al PSO, che presenta una maggior rapidità nella convergenza soprattutto durante le fasi iniziali.
Il WOA è invece caratterizzato da un andamento dell'ottimizzazione piuttosto regolare con scarsa capacità di fuoriuscire dai punti di minimo locale.
Il numero medio di valutazioni elevato del GA evidenzia inoltre una certa abilità nell'individuare soluzioni migliori con continuità.

Capitolo 4

Ricostruzione del campo di spostamenti di una piastra irrigidita in composito

In questo capitolo verrà discussa l'attività sperimentale eseguita. A seguito di un'ottimizzazione, effettuata mediante un algoritmo genetico, sono stati installati su di un pannello in composito irrigidito dei sensori di deformazione, la cui misura è stata fornita come input all'iFEM. Gli spostamenti ottenuti sono stati poi comparati con quelli sperimentali, valutati attraverso degli LVDT.
Inoltre, è stata eseguita un ulteriore ottimizzazione della disposizione presente sulla struttura, andando a ridurre il numero sia delle fibre che delle rosette.

4.1 Setup sperimentale

L'oggetto dell'investigazione sperimentale è un pannello alare, avente angolo di freccia pari a 3.93°, in composito, irrigidito mediante correnti a sezione a T. Le rispettive geometrie sono riportate nella Figura 4.1 ed in Figura 4.2.
Il multistrato è composto da lamine di prepreg TWILL T-300 avente le seguenti proprietà meccaniche.

E_{11} [GPa]	E_{22} [GPa]	ν_{12}	G_{12}= G_{23}= G_{13} [GPa]	Spessore [mm]
59.7	59.7	0.09	3.8	0.25

Tabella 4.1: Proprietà meccaniche TWILL T-300

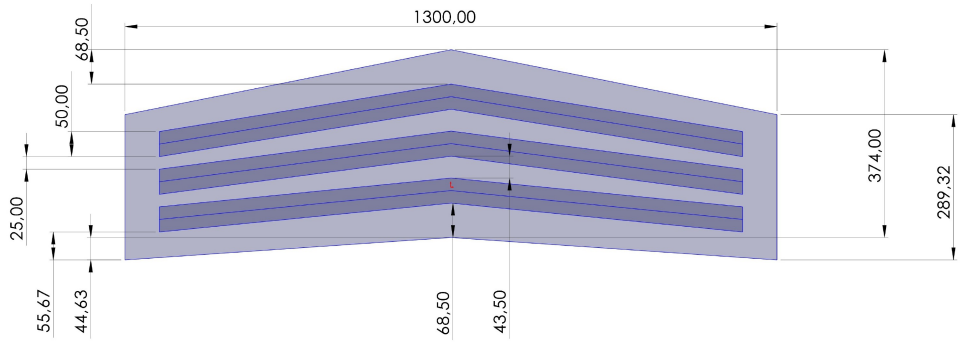

Figura 4.1: Geometria in pianta del pannello

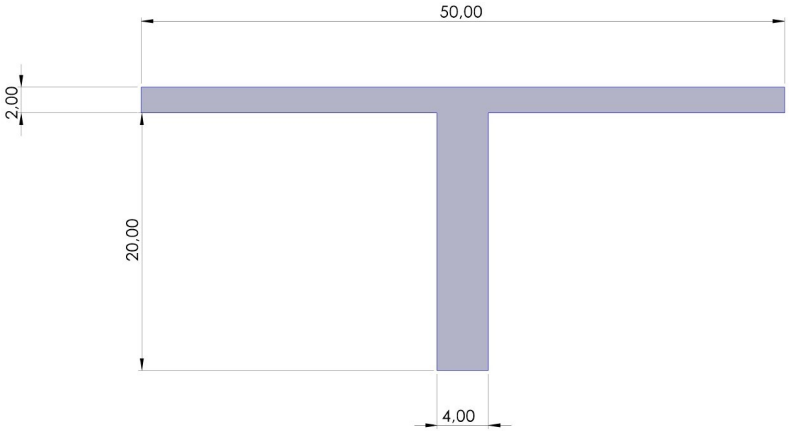

Figura 4.2: Sezione trasversale corrente

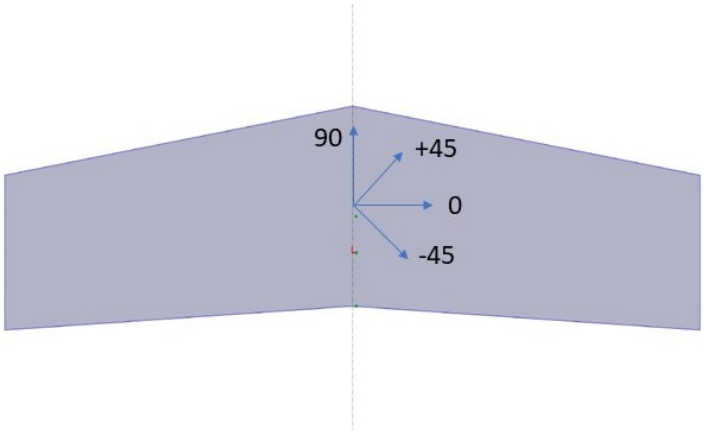

Figura 4.3: Direzioni fibre

La laminazione, sia del pannello che dei correnti, è simmetrica, con la seguente sequenza di stucking: $[45/0/0/45/0/0/0/45]_s$, rispetto la direzione di riferimento evidenziata in Figura 4.3. In particolare, le solette dei correnti sono state ottenute flettendo le plies delle anime di un angolo pari a 90°.
Il pannello è stato poi vincolato all'estremità attraverso dei semplici appoggi, realizzati tramite due barre semicilindriche di ferro, collocate a 41 mm dal tip di ciascuna semiala. In questo modo, viene permessa la sola rotazione attorno l'asse y del pannello.

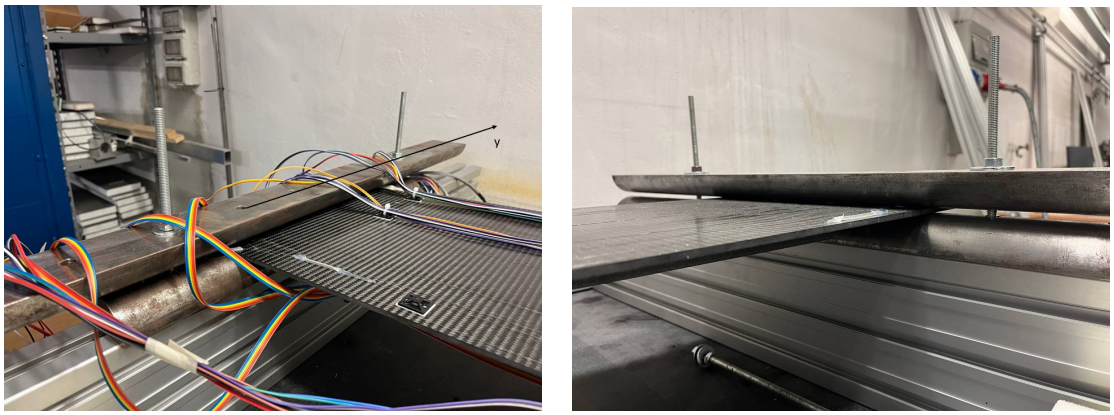

Figura 4.4: Semplice appoggio realizzato in laboratorio

Serrando due dadi, una barra trasversale spinge una sfera di ferro contro il pannello, la cui forma è stata scelta per poter trasferire il carico in un punto, come mostrato in Figura 4.5. Due celle di carico, installate alla base delle due barre filettate, permettono la misura del carico impartito.

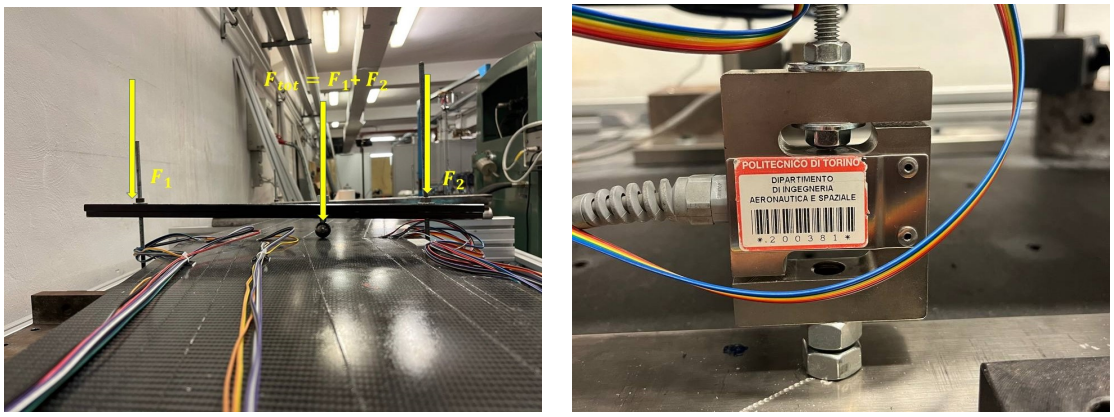

Figura 4.5: Sistema di carico

In particolare, il punto di applicazione è stato scelto per poter generare oltre che

una semplice flessione anche una torsione di natura geometrica, dato l'angolo di freccia del pannello.

4.2 Modelli numerici

Per individuare la disposizione finale dei sensori e per verificare i risultati finali sono stati implementati due diversi modelli numerici.
Per quanto riguarda il modello iFEM, sono stati utilizzati 672 iQS4 e 144 elementi beam inversi per un totale di 741 nodi e 4446 DOF, contando per ogni nodo sei gradi di libertà. In particolare, gli elementi shell inversi sono stati utilizzati con uno spessore di 4 mm per modellare lo skin del pannello ed uno spessore di 6 mm per tenere anche in conto lo spessore dei cap. In fine, l'anima dei correnti è stata modellata utilizzando gli elementi beam inversi considerando una sezione trasversale rettangolare di dimensioni $4\times$ 20 mm^2. Questo modello è servito sia all'individuazione della disposizione ottimale dei sensori sia alla ricostruzione del campo di spostamenti, utilizzando le deformazioni sperimentali misurate durante la prova.
Il modello FEM è formato invece da un totale di 3264 elementi shell, ottenuti dividendo in quattro elementi ciascun elemento del modello inverso, e 3407 nodi. I correnti sono stati modellati attraverso due elementi shell aventi 4 mm di spessore. In fine, così come per il modello iFEM gli elementi in comune con i cap sono stati modellati con shell di 6 mm. In Figura 4.6 è possibile osservare la mesh del modello utilizzato.
In particolare, gli spostamenti ottenuti sono stati utilizzati come riferimento per il calcolo dell'ERMS e quindi per il processo di ottimizzazione che ha permesso di identificare la configurazione da installare sulla struttura.

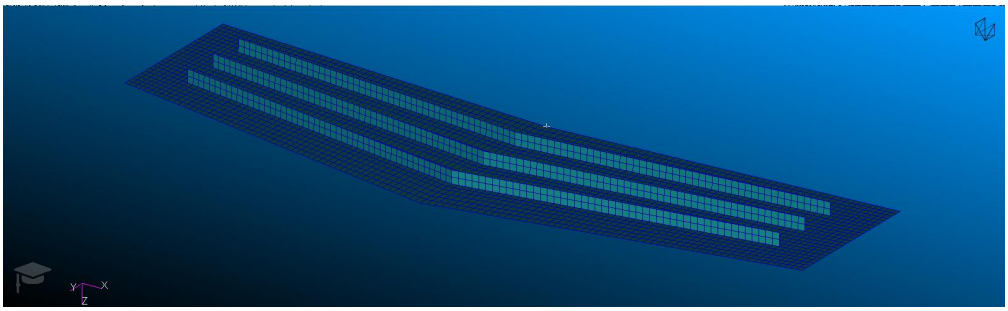

Figura 4.6: Modello FEM

4.3 Configurazione sensoristica

Sul pannello sono state disposte 7 fibre e 8 rosette nelle posizioni suggerite da un algoritmo genetico.

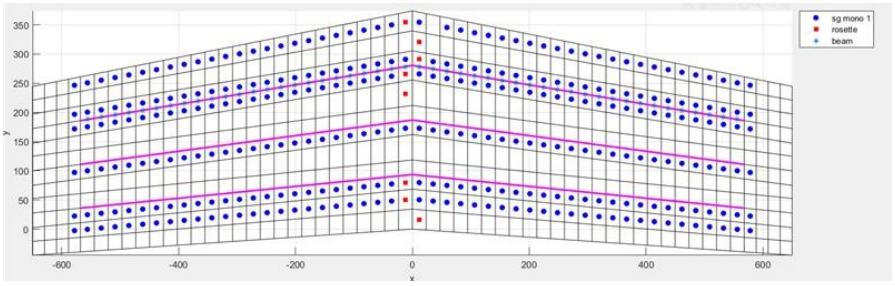

Figura 4.7: Disposizione dei sensori

Riguardo le fibre ottiche, la tecnologia adottata è la LUNA® high-definition distributed fiber optic strain sensing e permette la misura delle deformazioni lungo la direzione in cui la fibra viene collocata. Questa tipologia di sensore si basa sullo scattering di Rayleigh e sulla riflettometria nel dominio della frequenza ottica (OFDR).
La dispersione (scattering) è un cambiamento della direzione della radiazione propagata causato dalla non uniformità del materiale a livello molecolare: lo scattering di Rayleigh è dovuto all'interazione di onde elettromagnetiche con particelle di dimensioni minori rispetto alla loro lunghezza d'onda. Nel caso in questione, questi difetti sono rappresentati dalle impurità e disomogeneità presenti nel silicio amorfo che va a costituire il core ed il cladding della fibra. Tale fenomeno rientra nelle perdite di materiale e quindi determina la riduzione della potenza del segnale propagato.

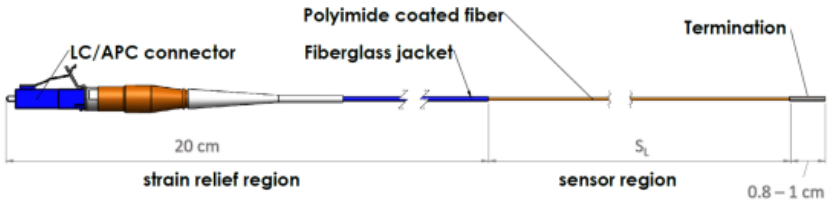

Figura 4.8: LUNA® high-definition distributed fiber optic strain sensing [39]

Circa l'OFDR, la sorgente del riflettometro è un laser che emette un segnale coerente la cui frequenza è variata linearmente in una larghezza di banda che si può estendere (in unità di lunghezza d'onda) di diverse decine di nanometri. Una

parte di questo segnale è immesso in ingresso nella fibra e il resto è usato come un oscillatore locale LO (local oscillator). Il segnale di backscattering è quindi unito al segnale LO e viene prodotta un'interferenza coerente che infine viene inviata nel ricevitore e misurata con tecniche interferometriche [40]: comparando il segnale con una misura di riferimento è poi possibile valutare le deformazioni.
In particolare, è stata utilizzata una fibra della lunghezza di 10.288 m con frequenza di misurazione pari a 1.04167 Hz e passo di 1.3 mm.

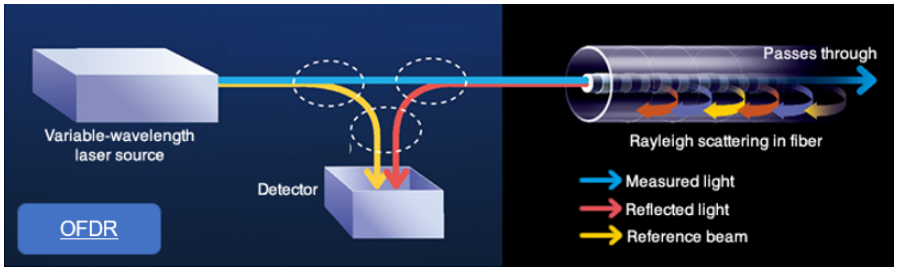

Figura 4.9: Schema del funzionamento di un OFDR [41]

Aspetto interessante riguarda il numero di misure di deformazione necessarie per la valutazione del campo di spostamenti attraverso l'iFEM. Lungo il pannello quest'ultime sono state disposte con una configurazione back-to-back: infatti, per un elemento inverse shell è possibile risalire alle deformazioni membranali e curvature sperimentali, mediante la sola conoscenza di due misure misure effettuate sul top e sul bottom dell'elemento, come è possibile osservare delle Equazioni 2.43 e 2.44.
Invece, per ogni corrente, sono state disposte tre fibre lungo l'anima, come evidenziato in Figura 4.10.

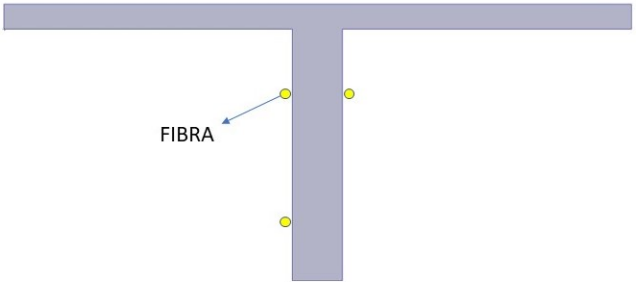

Figura 4.10: Disposizione delle fibre lungo l'anima dell'irrigidimento

Tale applicazione è stata necessaria in quanto per un elemento inverse beam di ordine zero, in particolare per l'elemento iTM2D0, sono necessarie 8 misure di deformazione per poter interpolare le otto deformazioni di sezione e_i $(i = 1, \ldots, 8)$ essendo e_1, e_4, e_5, e_6 costanti mentre e_2 ed e_3 lineari. Inoltre, sfruttando le due equazioni di equilibrio alla rotazione della trave che legano, ciascuna, un momento

flettente al corrispondente taglio e le relative equazioni costitutive si perviene alle seguenti equazioni

$$D_y e_{2,x} = G_z e_4 \qquad D_z e_{3,x} = G_y e_5 \tag{4.1}$$

In questo modo il numero di misure sperimentali si abbassa a sei, ottenibile attraverso tre fibre, in quanto le deformazioni di sezione e_4 e e_5 possono essere determinate analiticamente.
Infine, sono stati disposti in maniera casuale 6 LVDT, mostrati in Figura 4.11 per la valutazione dello spostamento al fine di poter poi effettuare il confronto con i risultati ottenuti dal modello agli elementi finiti.

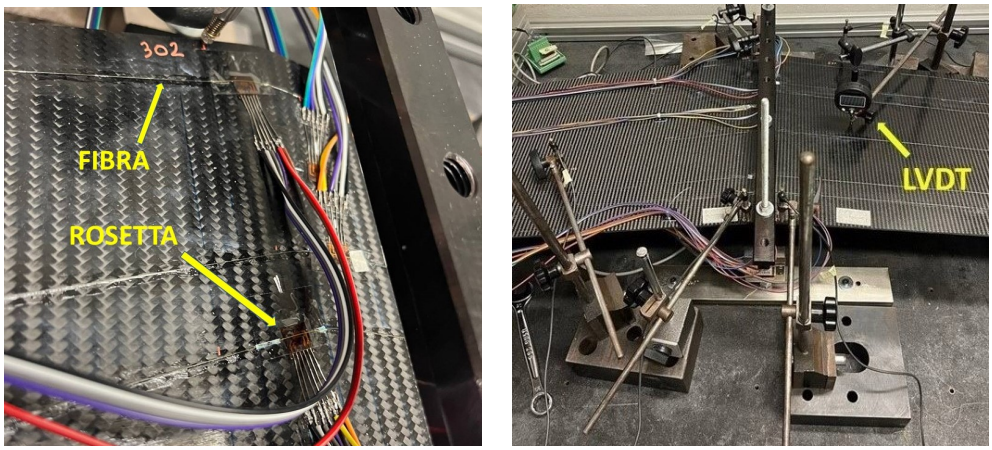

Figura 4.11: Dettaglio dei sensori utilizzati

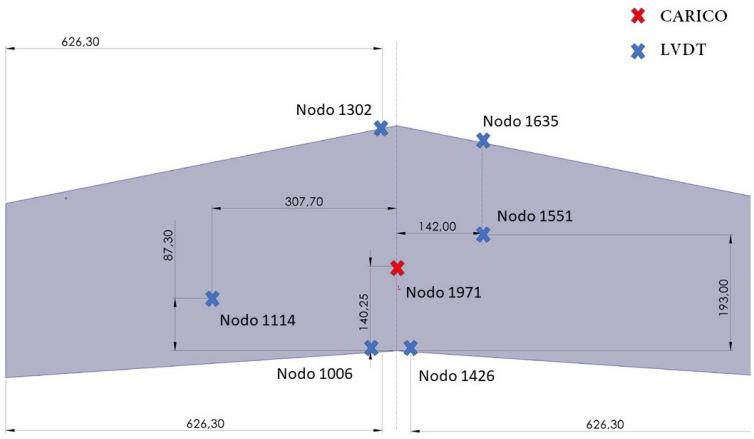

Figura 4.12: Posizione degli LVDT e del punto di applicazione del carico

4.4 Risultati sperimentali

4.4.1 Configurazione iniziale

Dai risultati riportati in Tabella 4.2 e Tabella 4.3 si deduce come l'accuratezza dell'iFEM risulta essere molto elevata con un errore relativo medio di circa il 5 %, pari a quello del FEM.

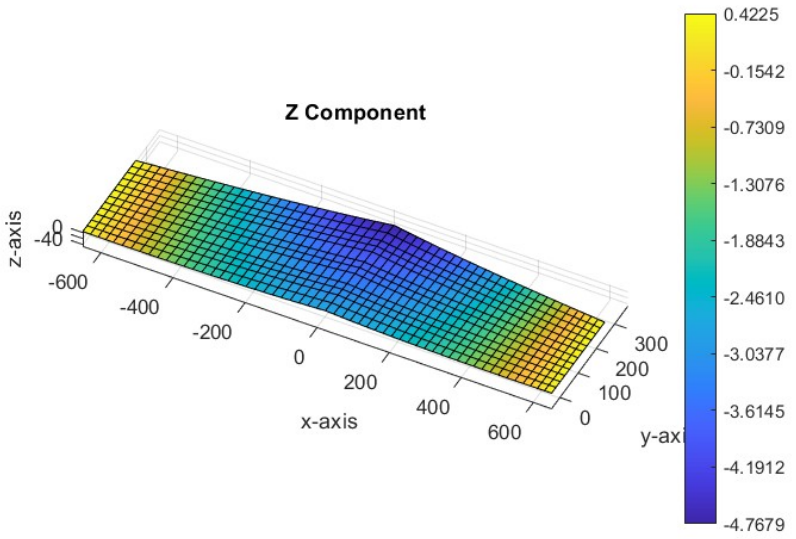

Figura 4.13: Campo di spostamenti: componente trasversale w

Nodo	Sperimentale [mm]	FEM [mm]	iFEM [mm]
1302	-5.121	-4.781 (6.64%)	-4.963 (3.07%)
1635	-4.542	-4.239 (6.65%)	-4.448 (2.05%)
1551	-3.731	-3.435 (7.92%)	-3.830 (2.66%)
1426	-2.846	-2.757 (3.11%)	-3.112 (9.33%)
1006	-2.877	-2.757 (4.19%)	-3.108 (7.98%)
1114	-2.311	-2.223 (3.78%)	-2.495 (7.97%)

Tabella 4.2: Spostamenti nei punti individuati

FEM [mm]	iFEM [mm]
5.384%	5.513%

Tabella 4.3: Media degli errori relativi: FEM e iFEM

4.4.2 Configurazioni con numero ridotto di sensori

Come si è potuto osservare, la ricostruzione del campo di spostamenti attraverso il metodo agli elementi finiti inversi risulta essere molto accurata. Chiaramente, la bontà del risultato finale è funzione sia del numero di sensori utilizzati che della disposizione di quest'ultimi. Un'ulteriore analisi eseguita è stata un'ottimizzazione condotta sulla configurazione di sensori già presente sul pannello. È stato definito come parametro di input un numero inferiore sia di fibre che di rosette al fine di verificare se si fosse potuto mantenere un livello di accuratezza accettabile o meno. Ciò non ha richiesto di eseguire ulteriori prove sperimentali in quanto si è lavorato con le deformazioni acquisite in precedenza.
Anzitutto è stata effettuata un analisi preliminare usando come set di deformazioni quelle ottenute dall'analisi agli elementi finiti. Vengono di seguito riportati i risultati ottenuti in termini di ERMS dello spostamento trasversale per le configurazioni scelte.

Numero di sensori	$ERMS$
5 fibre e 5 rosette	0.3745 %
5 fibre e 3 rosette	1.1598 %
3 fibre e 3 rosette	1.7947 %

Tabella 4.4: ERMS numerico dello spostamento trasversale delle tre configurazioni considerate

In particolare, per condurre le ottimizzazioni sono stati usati gli algoritmi di ottimizzazione BPSO e BWOA: entrambi hanno converso alla medesima disposizione e quindi allo stesso valore dell'ERMS, per tutti e tre i casi considerati. Ciò significa che le tre configurazioni individuate rappresentano, con elevata probabilità, il minimo assoluto per gli input presi in considerazione.

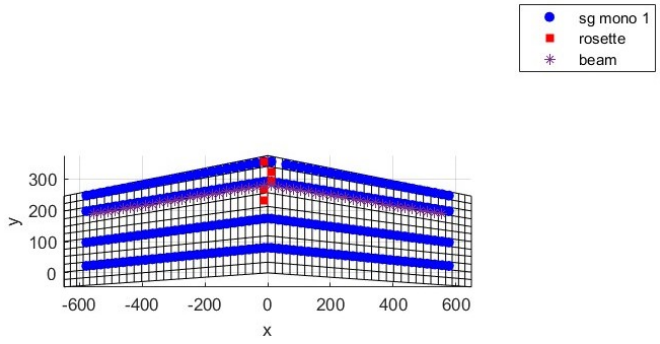

Figura 4.14: Configurazione ottimizzata: 5 fibre e 5 rosette

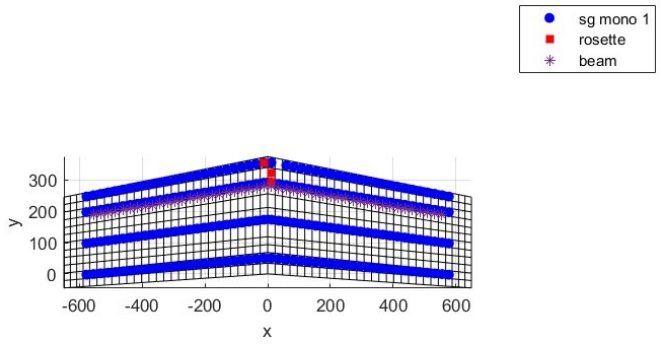

Figura 4.15: Configurazione ottimizzata: 5 fibre e 3 rosette

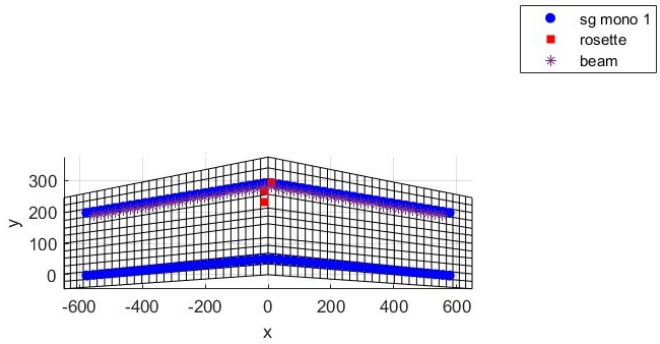

Figura 4.16: Configurazione ottimizzata: 3 fibre e 3 rosette

Vengono riportati in Tabella 4.5 gli spostamenti numerici ottenuti mediante iFEM per tutte e tre le configurazioni individuate.
Dai risultati ottenuti è possibile osservare come la disposizione dei sensori risulta

Nodo	Exp [mm]	iFEM [mm] 7 f. - 8 r.	iFEM [mm] 5 f. - 5 r.	iFEM [mm] 3 f. - 5 r.	iFEM [mm] 3 f. - 3 r.
1302	-5.121	-4.963 (3.07%)	-5.080 (0.81%)	-4.995 (2.45%)	-5.005 (2.27%)
1635	-4.542	-4.448 (2.05%)	-4.541 (0.023%)	-4.474 (1.50%)	-4.528 (0.297%)
1551	-3.731	-3.830 (2.66%)	-3.803 (1.93%)	-3.770 (1.23%)	-3.737 (0.186%)
1426	-2.846	-3.112 (9.33%)	-3.133 (10.1%)	-3.120 (9.61%)	-3.094 (8.68%)
1006	-2.877	-3.108 (7.98%)	-3.132 (8.00%)	-3.120 (8.40%)	-3.092 (7.43%)
1114	-2.311	-2.495 (7.97%)	-2.482 (7.39%)	-2.478 (7.24%)	-2.466 (6.72%)

Tabella 4.5: Spostamenti nei punti individuati

iFEM 7 f. - 8 r.	iFEM 5 f. - 5 r.	iFEM 3 f. - 5 r.	iFEM 3 f. - 3 r.
5.51%	4.84%	5.07 %	4.26 %

Tabella 4.6: Media degli errori relativi: FEM e iFEM

essere determinante per l'iFEM: infatti, dalla Tabella 4.6 si nota come la configurazione avente tre fibre e tre rosette presenta una media dell'errore relativo inferiore rispetto alle altre. Chiaramente, da un punto di vista sperimentale queste variazioni sono poco significative ma è possibile dedurre che, se non correttamente disposti, sensori aggiuntivi determinano una perdita di precisione nel calcolo della risposta della struttura.

Capitolo 5

Conclusioni e sviluppi futuri

In questo lavoro di tesi sono stati sviluppati e testati due algoritmi di ottimizzazione metaeuristici, rispettivamente il PSO ed il WOA, come alternativa all'ormai noto ed ampiamente utilizzato algoritmo genetico con lo scopo di verificare se, nelle loro forme continue e binarie, potessero essere comparabili se non superiori in termini di velocità di convergenza ed accuratezza del risultato finale.
La motivazione per la scelta del PSO risiede nel fatto che, in letteratura, risulta essere uno dei principali framework utilizzati per la ricerca della disposizione ottimale di sensori, sia in campo strutturale che per la copertura di sensori wireless impiegati in diversi contesti: industriale, agricolo, ambientale e medico. Invece il WOA, oltre ad essere una delle tecniche di swarm intelligence più recenti, offre una valida alternativa in termini computazionali [28] e capacità di fuoriuscire da possibili punti di minimo locale.
Dai test eseguiti si evince come i due algoritmi risultano essere una valida alternativa al GA nel momento in cui la complessità del problema è minima ovvero quando il possibile numero di configurazioni è limitato: infatti, sia in termini di minor numero di valutazioni dell'iFEM che di miglior valore dell'ERMS, sono proprio il PSO ed il WOA ad ottenere i migliori risultati come visto nel Capitolo 3. L'algoritmo genetico invece sembra essere la scelta migliore quando il numero delle possibili soluzioni risulta estremamente elevato.
Una possibile spiegazione è da individuare nel modo in cui i tre algoritmi lavorano. Il GA effettua le varie operazioni di ricerca "ragionando" in maniera del tutto binaria: le varie operazioni di mutazione, inversione o cross-over vengono eseguite operando direttamente sui singoli bit. Ciò sembrerebbe garantire una maggiore capacità di esplorazione dello spazio di ricerca. Di contro, il PSO ed il WOA utilizzano lo strumento della funzione di trasferimento. Le equazioni sulle quali sono costruiti forniscono infatti valori decimali che vengono dati come input alla funzione di trasferimento: l'output viene quindi confrontato con un valore randomico ed a seguito della legge implementata il singolo bit può permanere nel suo stato o variare.

Dunque, la dinamica con la quale la ricerca viene effettuata è completamente diversa e trattandosi di un problema discreto, il modo in cui opera il GA risulta essere più adeguato.
Alcuni possibili miglioramenti vengono discussi brevemente di seguito.
Come detto in precedenza, è stata utilizzata una codifica in binario a nove bit per poter rappresentare gli individui delle metaeuristiche considerate. Aspetto negativo è che con tale numero non è possibile rappresentare tutte le possibili posizioni delle rosette ed è necessario ridurre lo spazio di ricerca con cognizione di causa. Una possibile modifica potrebbe consistere nell'utilizzare una codifica a 10 bit: in questo modo si avrebbe la possibilità di codificare i valori numerici che vanno dall'1 al 1023, coprendo dunque l'intero dominio della funzione obiettivo. Ovviamente, si avrebbero delle codifiche aggiuntive da dover gestire in maniera opportuna associando magari a ciascun elemento del dominio altre conversioni in binario. Oppure, si potrebbe completamente modificare il paradigma di codifica affidando ad ogni possibile posizione il valore di un bit per esprimere la relativa sensorizzazione e del suo complemento per l'assenza del posizionamento del sensore.
Inoltre, le metaeuristiche sono per definizione delle strutture di ottimizzazione che possono essere combinate fra di loro. Dal momento che, per le ottimizzazioni più complesse, il GA sembra essere la scelta migliore, si potrebbe pensare di introdurre all'interno del WOA o del PSO le strategie di cross-over o mutazione, per poter conferire una maggiore eterogeneità all'interno della popolazione e quindi aumentare le performance in termini di esplorazione dello spazio di ricerca.

Appendice A

Funzioni di forma elemento iTM2D0

Vengono di seguito riportate le funzioni di forma per un elemento iTM2D0

$$[L_1^{(1)}, L_2^{(1)}] = \frac{1}{2}[(1-\zeta), \frac{1}{2}(1+\zeta)] \tag{A.1}$$

$$[L_1^{(2)}, L_r^{(3)}, L_2^{(4)}] = \frac{1}{2}[\zeta(\zeta-1), 2(1-\zeta^2), \zeta(\zeta+1)] \tag{A.2}$$

$$[L_1^{(4)}, L_2^{(4)}] = \frac{1}{6}\zeta(4\zeta^2-1)[(\zeta-1), (1+\zeta)] \tag{A.3}$$

$$[L_1^{(6)}, L_r^{(6)}, L_2^{(6)}] = \frac{1}{3}(1-\zeta^2)[4\zeta(2\zeta-1), 3(1-4\zeta^2), 4\zeta(2\zeta+1)] \tag{A.4}$$

$$[N_1^{(3)}, N_r^{(3)}, N_2^{(3)}] = \frac{L^e}{24}(1-\zeta^2)[(2\zeta-3), -4\zeta, (2\zeta+3)] \tag{A.5}$$

$$\begin{aligned}[\bar{N}_1^{(3)}, \bar{N}_q^{(3)}, \bar{N}_3^{(3)}, \bar{N}_s^{(3)}, \bar{N}_2^{(3)}] =& \frac{4}{3L^e}(1-\zeta^2)[(4\zeta-3), -2(8\zeta-3), \\ & 24\zeta - 2(8\zeta+3), (4\zeta+3)]\end{aligned} \tag{A.6}$$

dove $\zeta = 2x/L^e - 1 \in [-1,1]$, $x \in [0, L^e]$. In particolare, i pedici 1 e 2 rappresentano la numerazione due due nodi dell'elemento mentre q, r, e s indicano i nodi equispaziati interni all'elemento.

Appendice B

Funzioni di forma elemento iQS4

Vengono di seguito riportate le funzioni di forma per un elemento iQS4

$$N_1 = \frac{(1-s)(1-t)}{4} \tag{B.1}$$

$$N_2 = \frac{(1+s)(1-t)}{4} \tag{B.2}$$

$$N_3 = \frac{(1+s)(1+t)}{4} \tag{B.3}$$

$$L_1 = y_{14}N_8 - y_{21}N_5 \tag{B.4}$$

$$L_2 = y_{21}N_5 - y_{32}N_6 \tag{B.5}$$

$$L_3 = y_{32}N_6 - y_{43}N_7 \tag{B.6}$$

$$L_4 = y_{43}N_7 - y_{14}N_8 \tag{B.7}$$

$$M_1 = x_{41}N_8 - x_{12}N_5 \tag{B.8}$$

$$M_2 = x_{12}N_5 - x_{23}N_6 \tag{B.9}$$

$$M_3 = x_{23}N_6 - x_{34}N_7 \tag{B.10}$$

$$M_4 = x_{34}N_7 - x_{41}N_8 \tag{B.11}$$

dove

$$N_5 = \frac{(1-s^2)(1-t)}{4} \tag{B.12}$$

$$N_6 = \frac{(1+s)(1-t^2)}{4} \tag{B.13}$$

$$N_7 = \frac{(1-s^2)(1+t)}{4} \tag{B.14}$$

$$N_8 = \frac{(1-s)(1-t^2)}{4} \tag{B.15}$$

Le funzioni di forma sono espresse in funzione delle coordinate locali dei nodi dell'elemento

$$x_{ij} = x_i - x_j \tag{B.16}$$

$$y_{ij} = y_i - y_j \tag{B.17}$$

con i = 1,2,3,4 e j = 1,2,3,4 e delle coordinate naturale s e $t \in$ [-1,1].

Appendice C

Risultati ottimizzazione per la disposizione di fibre e rosette

Di seguito sono riportati gli andamenti della miglior fitness individuata in funzione del numero di iterazioni delle dieci analisi per i due casi presi in esame.

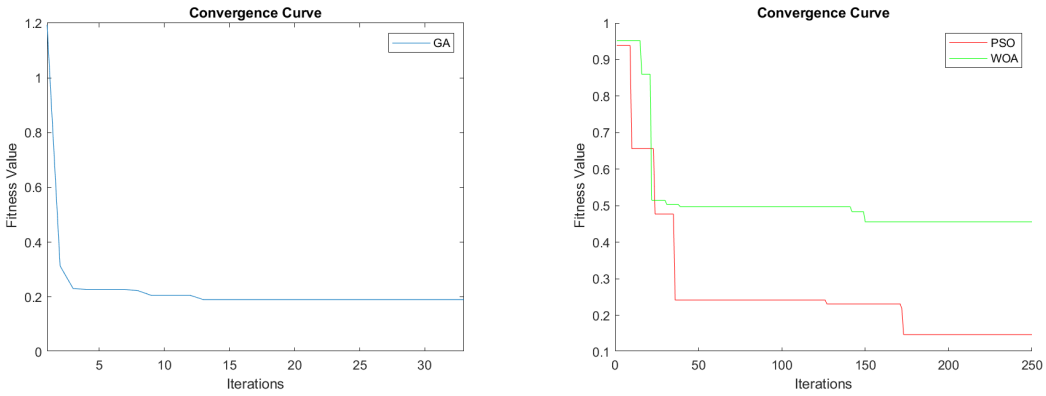

Figura C.1: Run 1: 50 individui GA e 28 individui PSO/WOA

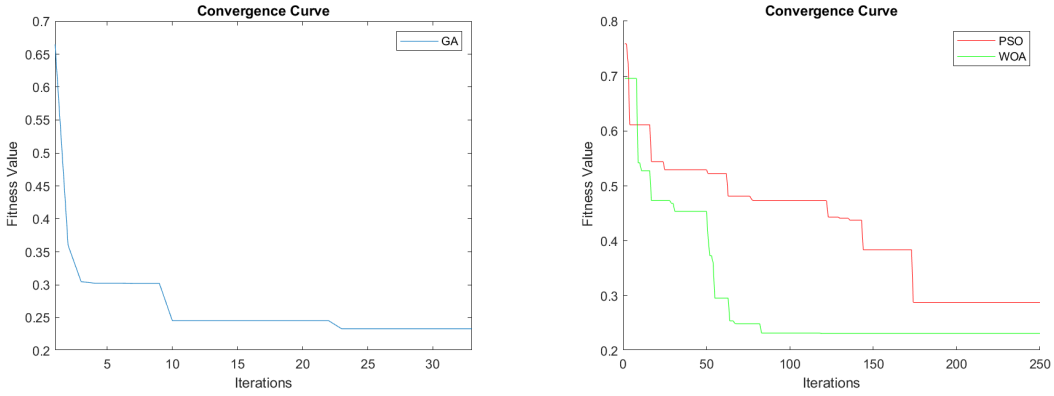

Figura C.2: Run 2: 50 individui GA e 28 individui PSO/WOA

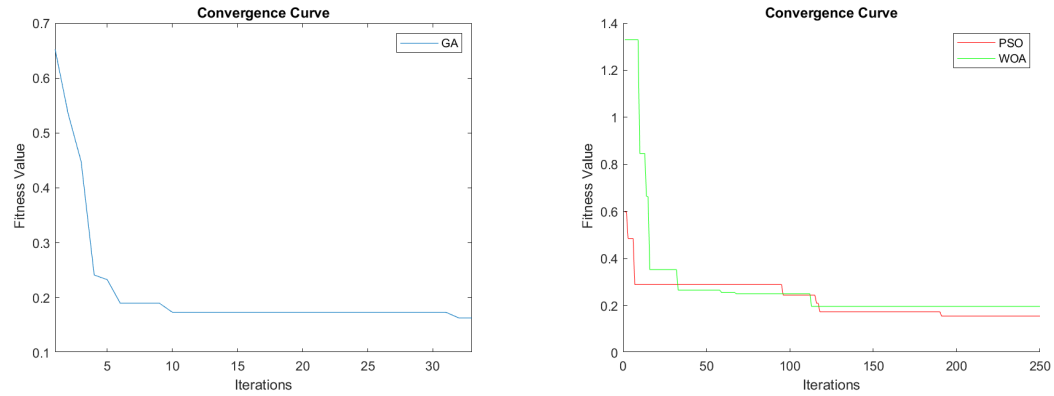

Figura C.3: Run 3: 50 individui GA e 28 individui PSO/WOA

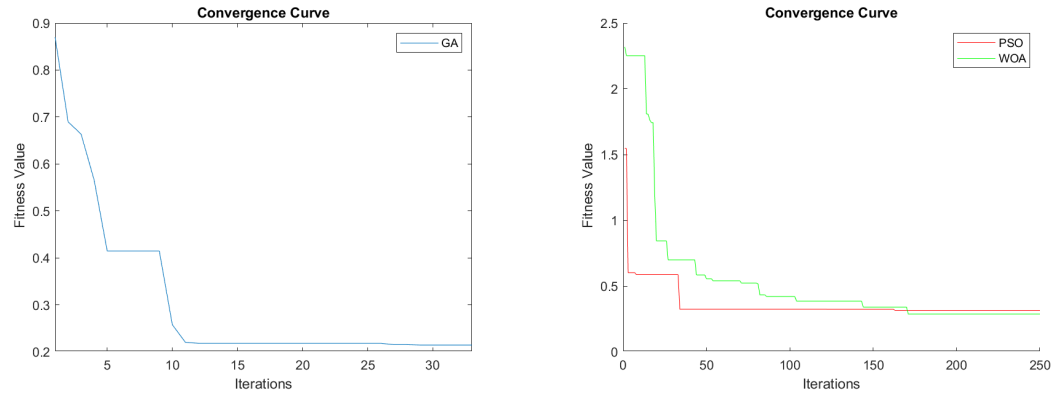

Figura C.4: Run 4: 50 individui GA e 28 individui PSO/WOA

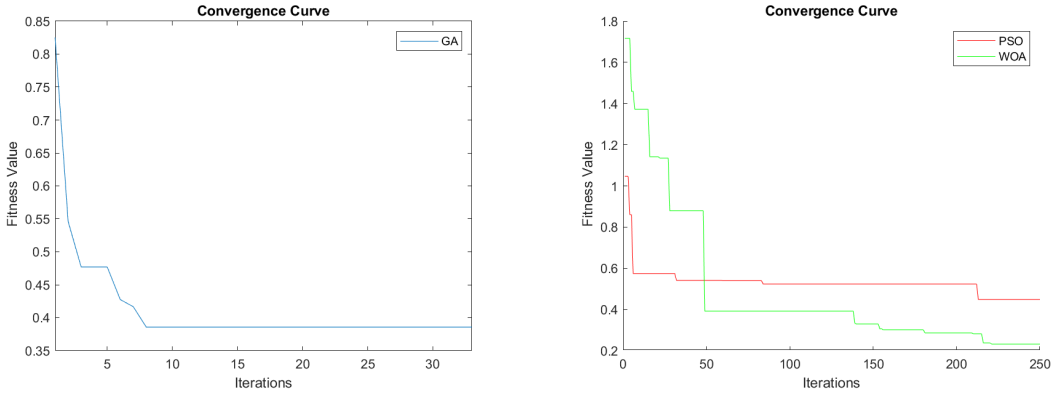

Figura C.5: Run 5: 50 individui GA e 28 individui PSO/WOA

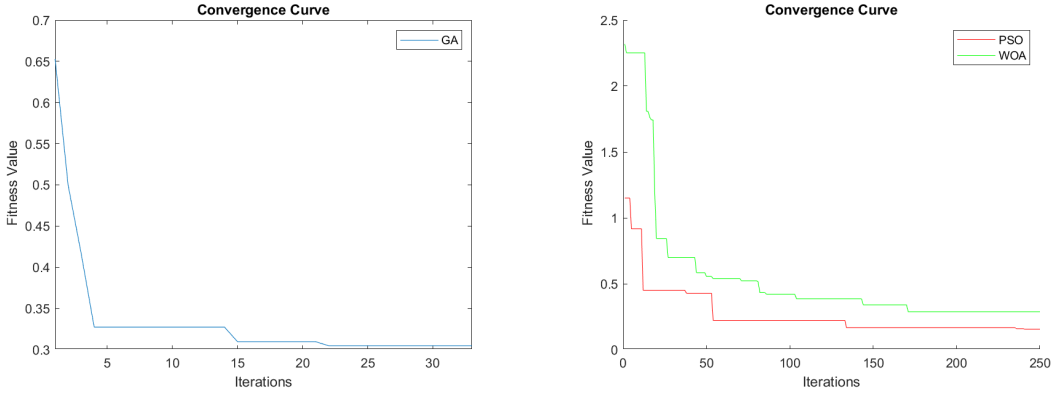

Figura C.6: Run 6: 50 individui GA e 28 individui PSO/WOA

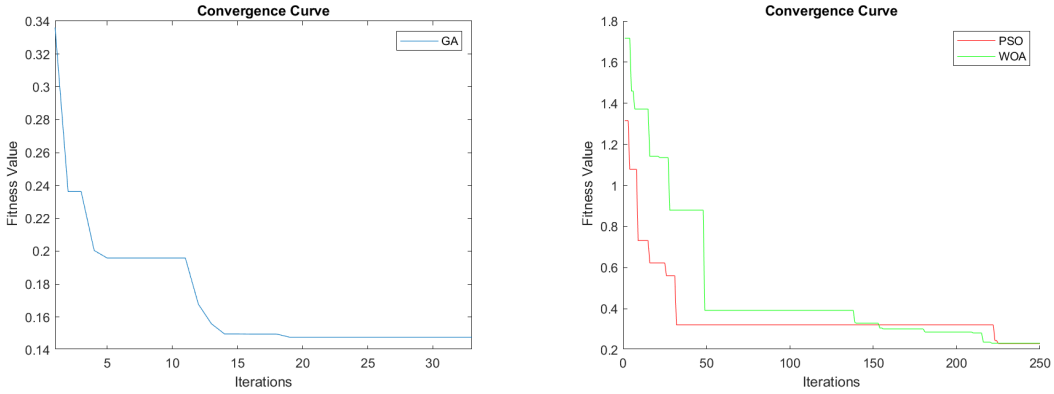

Figura C.7: Run 7: 50 individui GA e 28 individui PSO/WOA

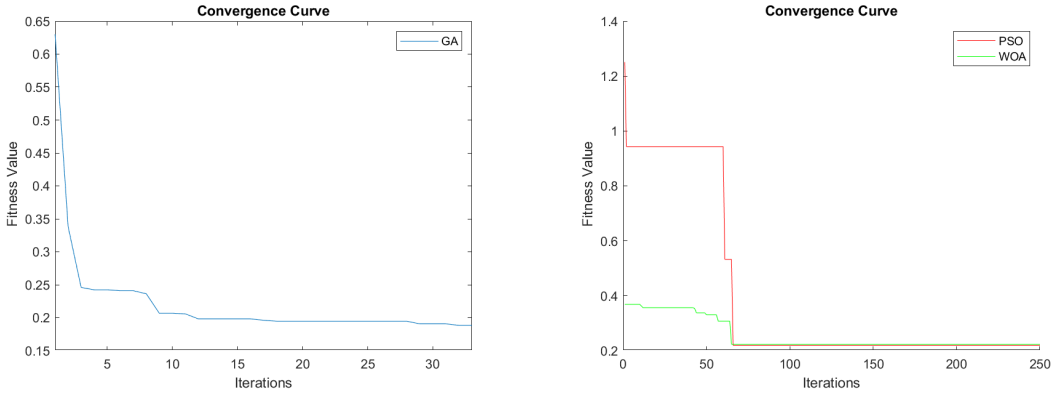

Figura C.8: Run 8: 50 individui GA e 28 individui PSO/WOA

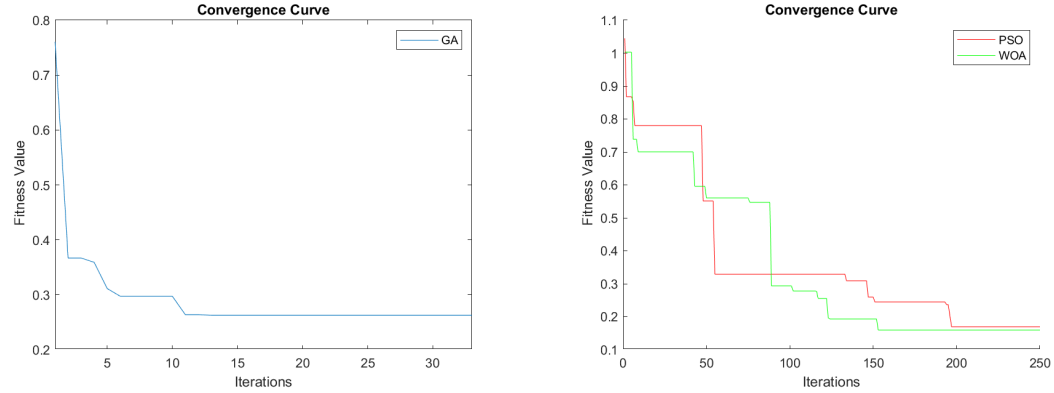

Figura C.9: Run 9: 50 individui GA e 28 individui PSO/WOA

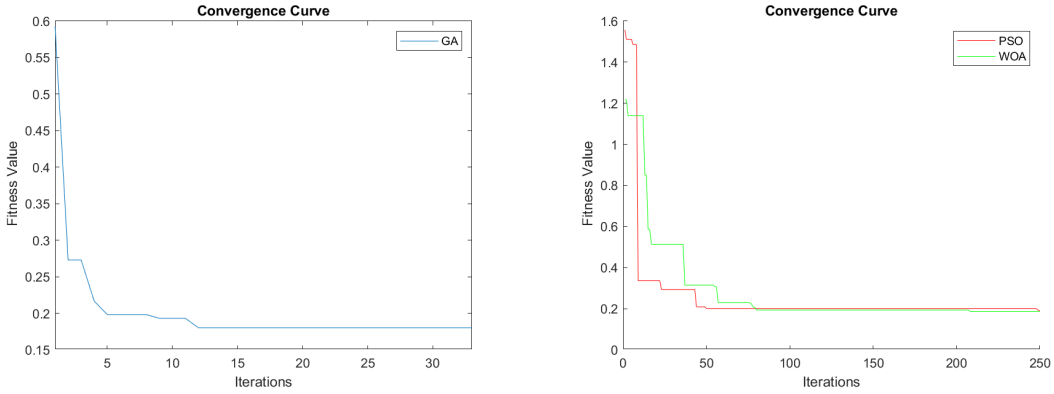

Figura C.10: Run 10: 50 individui GA e 28 individui PSO/WOA

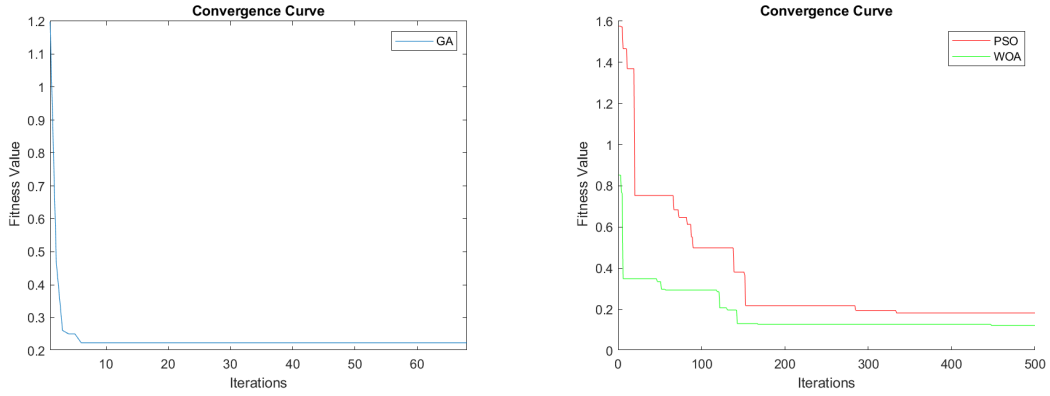

Figura C.11: Run 1: 100 individui GA e 14 individui PSO/WOA

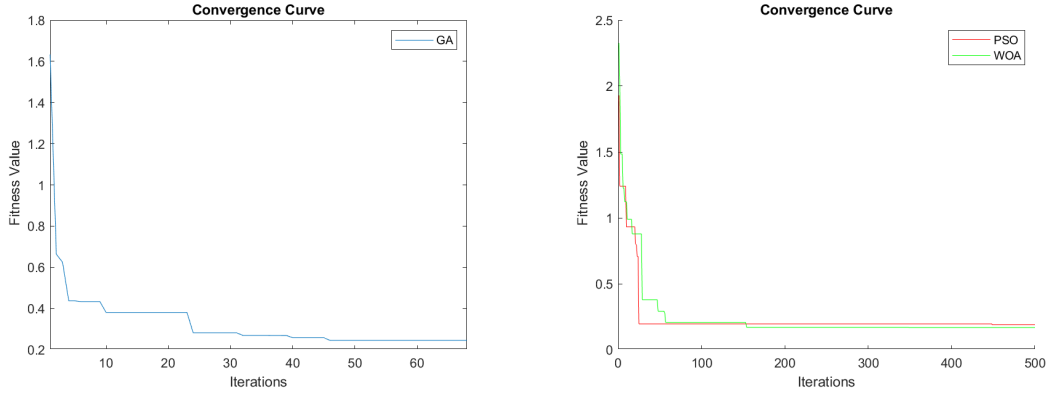

Figura C.12: Run 2: 100 individui GA e 14 individui PSO/WOA

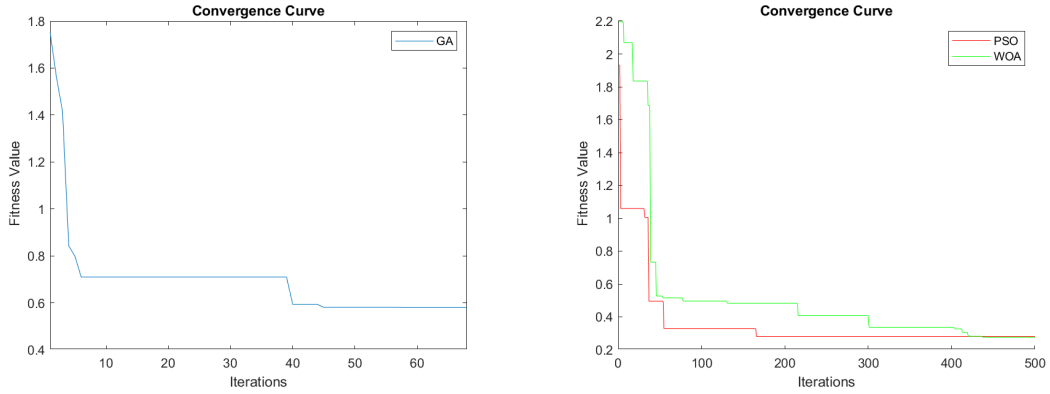

Figura C.13: Run 3: 100 individui GA e 14 individui PSO/WOA

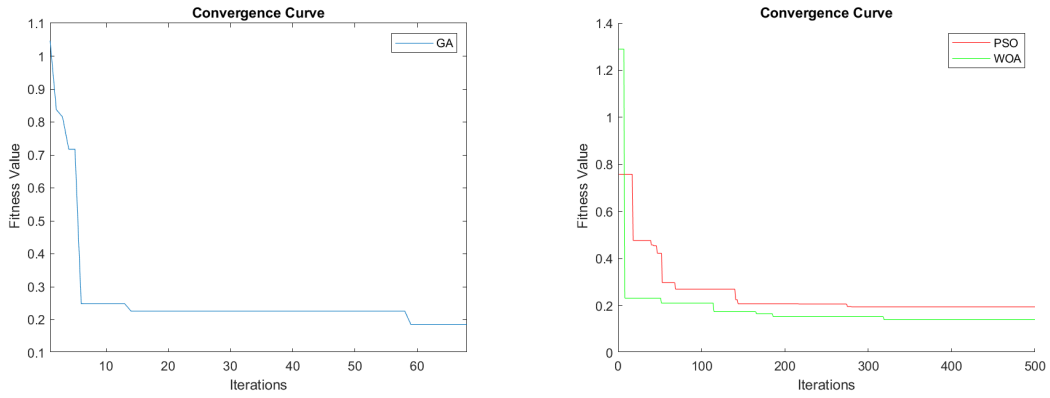

Figura C.14: Run 4: 100 individui GA e 14 individui PSO/WOA

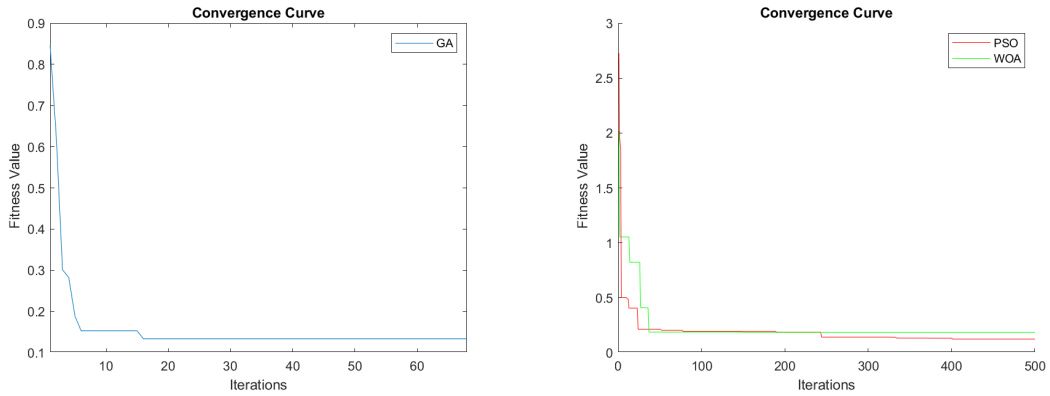

Figura C.15: Run 5: 100 individui GA e 14 individui PSO/WOA

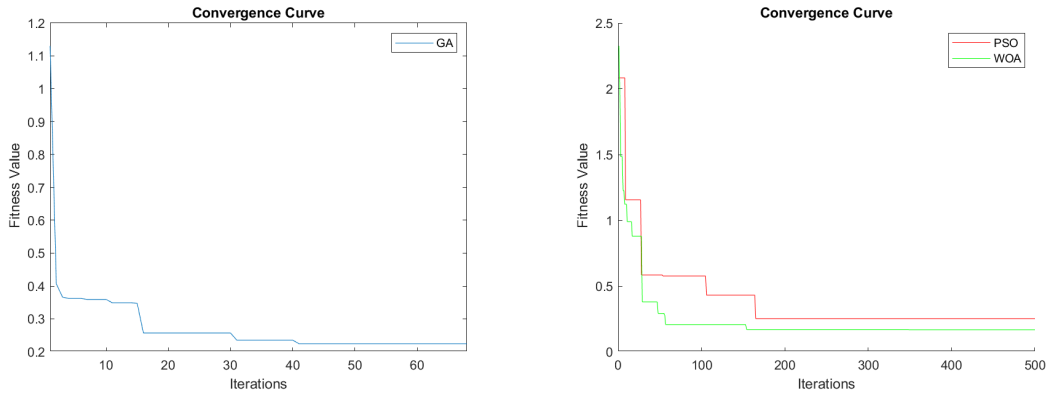

Figura C.16: Run 6: 100 individui GA e 14 individui PSO/WOA

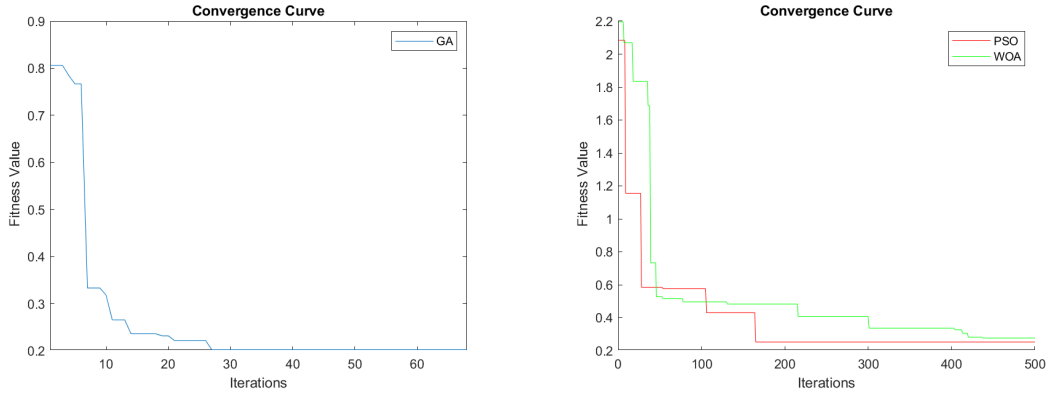

Figura C.17: Run 7: 100 individui GA e 14 individui PSO/WOA

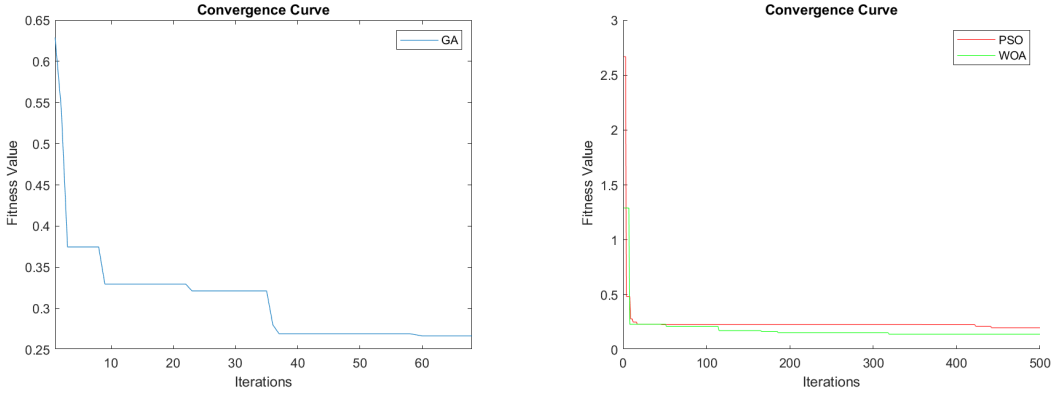

Figura C.18: Run 8: 100 individui GA e 14 individui PSO/WOA

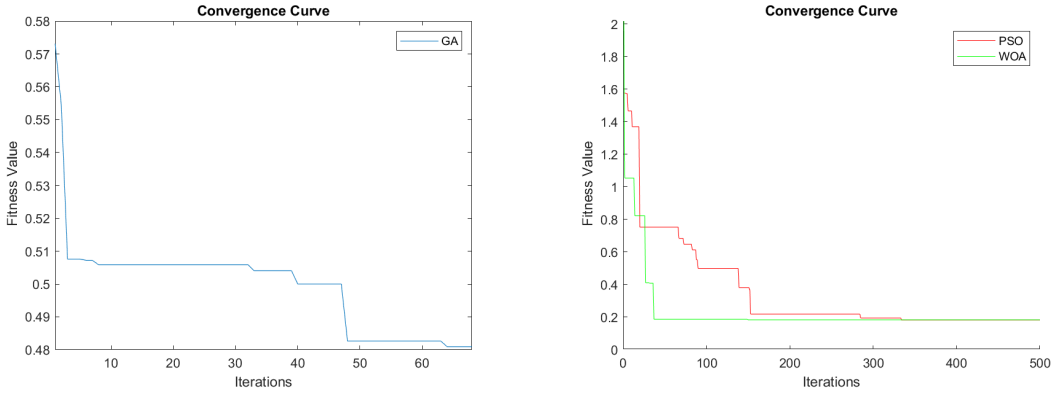

Figura C.19: Run 9: 100 individui GA e 14 individui PSO/WOA

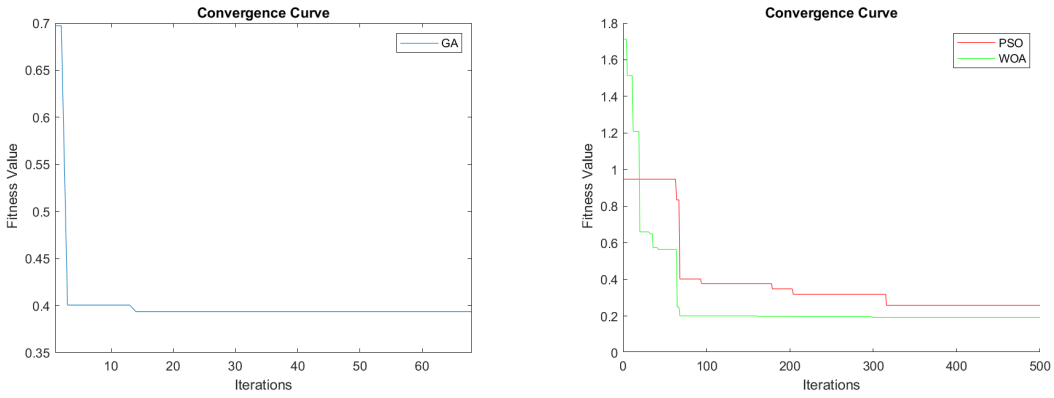

Figura C.20: Run 10: 100 individui GA e 14 individui PSO/WOA

Di seguito sono invece riportati i risultati delle 10 run in termini di best fitness, di iterazioni effettuate per individuarla e numero di valutazioni iFEM necessarie con 14 individui per quanto riguarda il WOA ed il PSO e 100 per il GA e le configurazioni di sensori ottenute con l'ottimizzazione eseguita con il PSO ed il WOA.

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.1826	334	4676
2	0.1959	25	350
3	0.2813	166	2324
4	0.2064	275	3850
5	0.1873	37	518
6	0.2523	165	2310
7	0.2523	165	2311
8	0.1986	442	6188
9	0.1811	334	4676
10	0.2585	316	4424

Tabella C.1: Risultati PSO: 14 individui e 500 iterazioni

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.1314	143	2002
2	0.1691	154	2156
3	0.2767	437	6118
4	0.1401	319	4466
5	0.1236	401	5614
6	0.1714	154	2156
7	0.2743	437	6118
8	0.1423	319	4466
9	0.1874	37	518
10	0.1932	68	952

Tabella C.2: Risultati WOA: 14 individui e 500 iterazioni

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.1905	13	2600
2	0.2331	23	4600
3	0.1631	32	6400
4	0.2150	27	5400
5	0.3857	8	1600
6	0.3046	22	4400
7	0.1476	19	3800
8	0.1883	32	6400
9	0.2630	11	2200
10	0.1801	12	2400

Tabella C.3: Risultati GA: 100 individui e 33 iterazioni

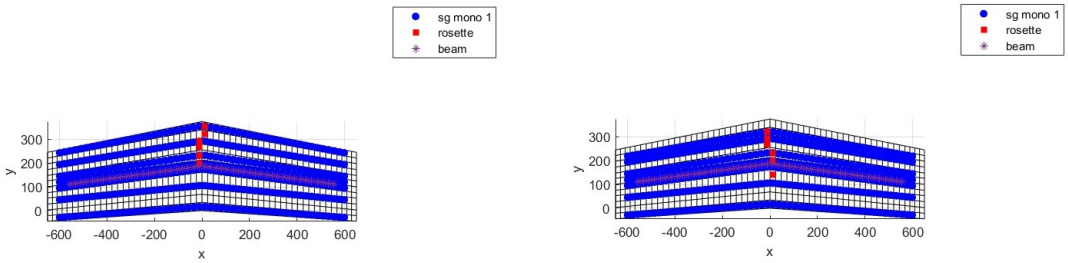

Figura C.21: Run 1 con 14 individui: PSO e WOA

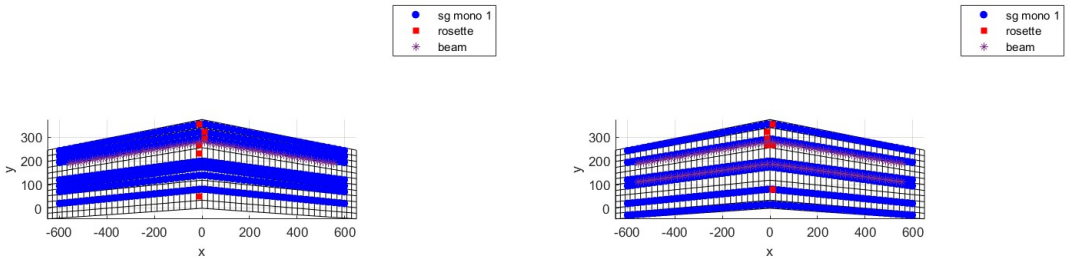

Figura C.22: Run 2 con 14 individui: PSO e WOA

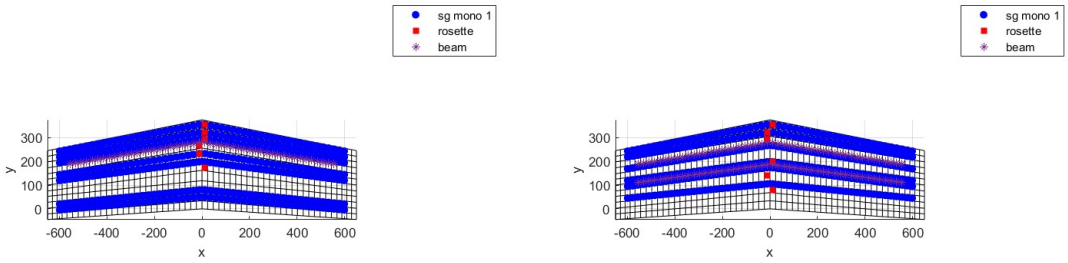

Figura C.23: Run 3 con 14 individui: PSO e WOA

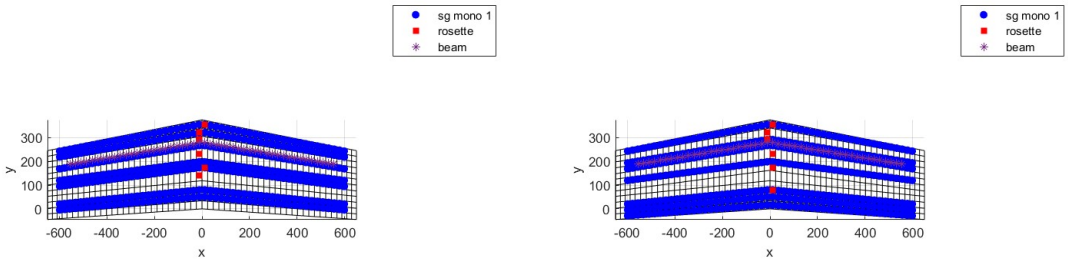

Figura C.24: Run 4 con 14 individui: PSO e WOA

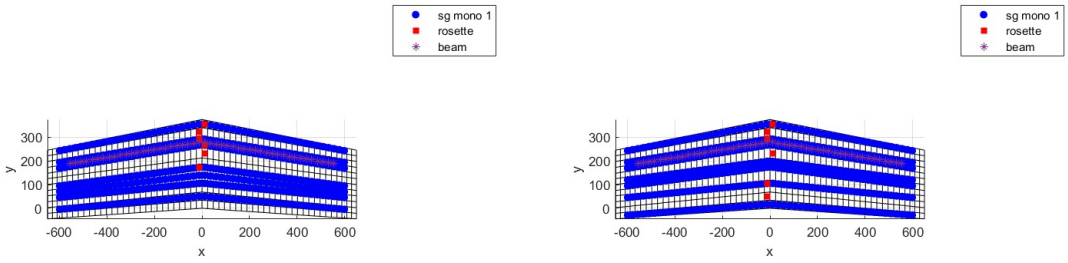

Figura C.25: Run 5 con 14 individui: PSO e WOA

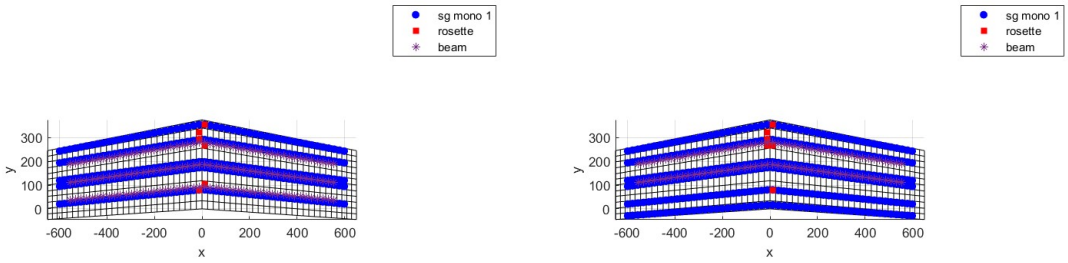

Figura C.26: Run 6 con 14 individui: PSO e WOA

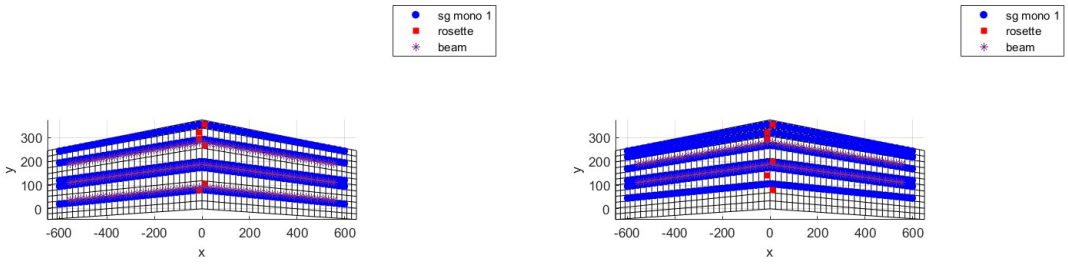

Figura C.27: Run 7 con 14 individui: PSO e WOA

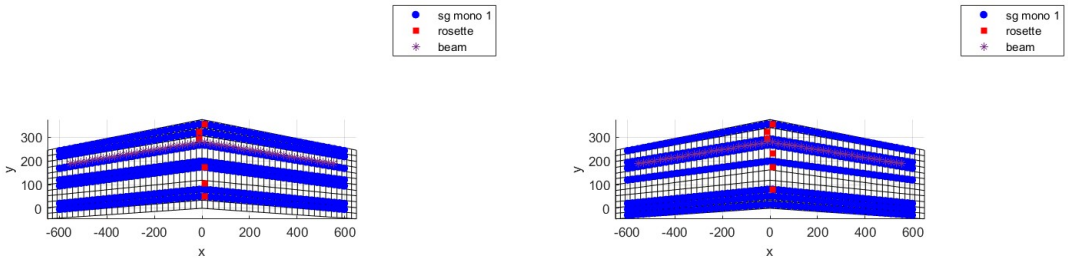

Figura C.28: Run 8 con 14 individui: PSO e WOA

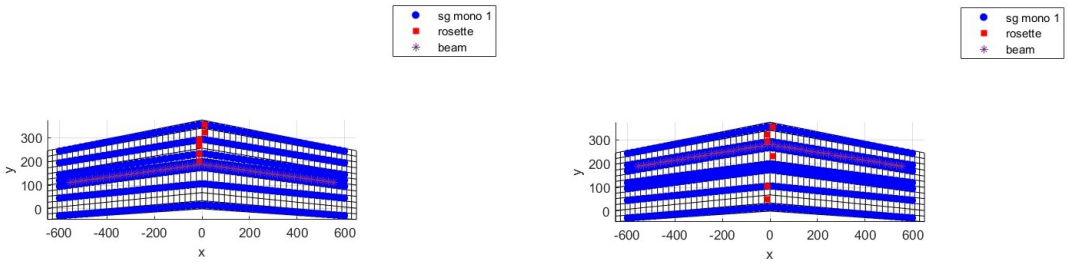

Figura C.29: Run 9 con 14 individui: PSO e WOA

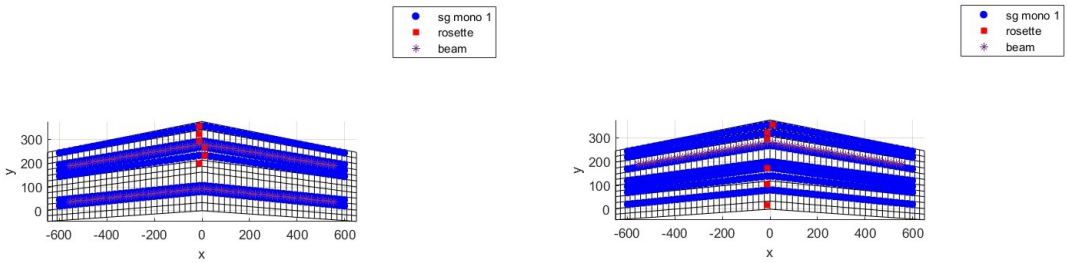

Figura C.30: Run 10 con 14 individui: PSO e WOA

Di seguito sono invece riportati i risultati delle 10 run in termini di best fitness, di iterazioni effettuate per individuarla e numero di valutazioni iFEM necessarie con 28 individui per quanto riguarda il WOA ed il PSO e 50 per il GA e le configurazioni di sensori ottenute con l'ottimizzazione eseguita con il PSO ed il WOA.

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.1471	173	4844
2	0.2879	174	4872
3	0.1555	191	5348
4	0.3155	163	4564
5	0.4189	213	5964
6	0.1545	241	6748
7	0.2305	225	6300
8	0.2194	66	1848
9	0.1693	197	5516
10	0.2010	50	1400

Tabella C.4: Risultati PSO 28: individui e 250 iterazioni

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.4556	150	4200
2	0.2320	83	2324
3	0.1966	113	3164
4	0.2866	171	4788
5	0.2322	221	6188
6	0.2864	178	4788
7	0.2321	265	6188
8	0.2128	65	1820
9	0.1595	153	4284
10	0.1932	80	2240

Tabella C.5: Risultati WOA 28: individui e 250 iterazioni

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	0.2239	6	600
2	0.2448	46	4600
3	0.5804	45	4500
4	0.1850	59	5900
5	0.1330	16	1600
6	0.2245	41	4100
7	0.2012	27	2700
8	0.2666	60	6000
9	0.4810	64	6400
10	0.3932	14	1400

Tabella C.6: Risultati GA: 50 individui e 68 iterazioni

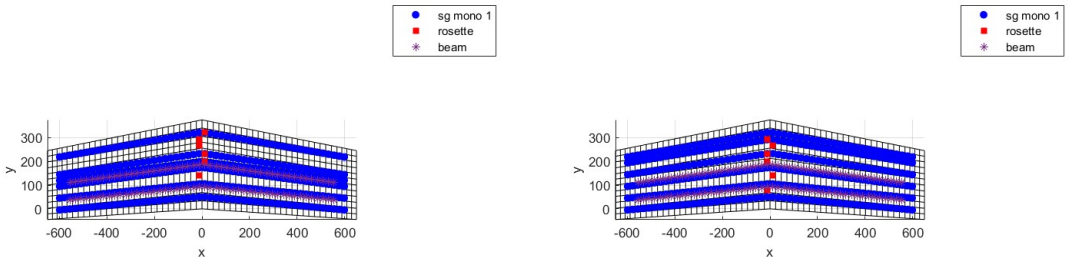

Figura C.31: Run 1 con 28 individui: PSO e WOA

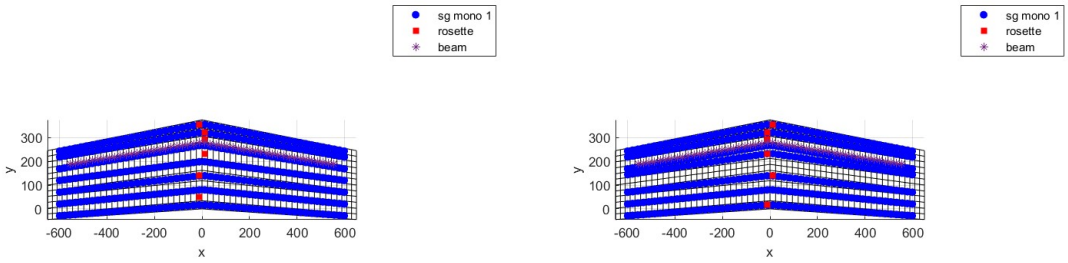

Figura C.32: Run 2 con 28 individui: PSO e WOA

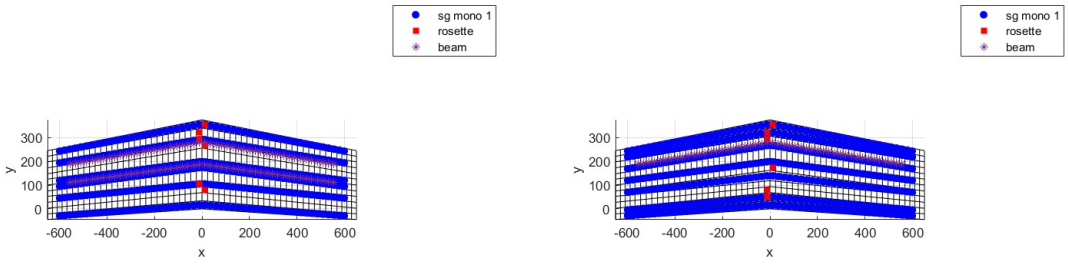

Figura C.33: Run 3 con 28 individui: PSO e WOA

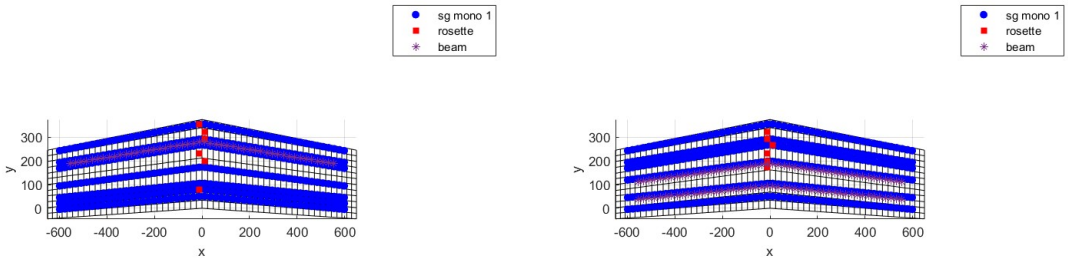

Figura C.34: Run 4 con 28 individui: PSO e WOA

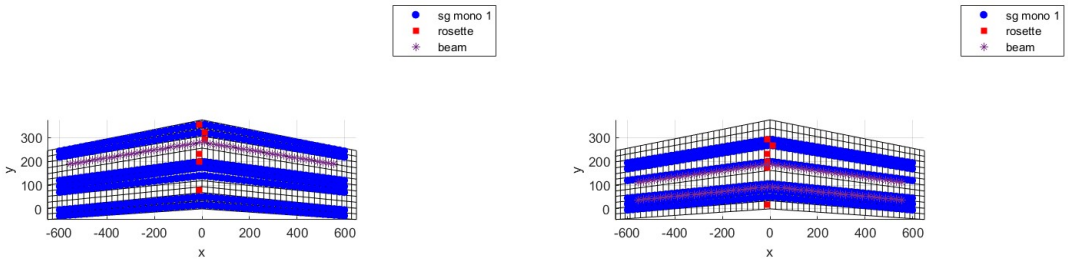

Figura C.35: Run 5 con 28 individui: PSO e WOA

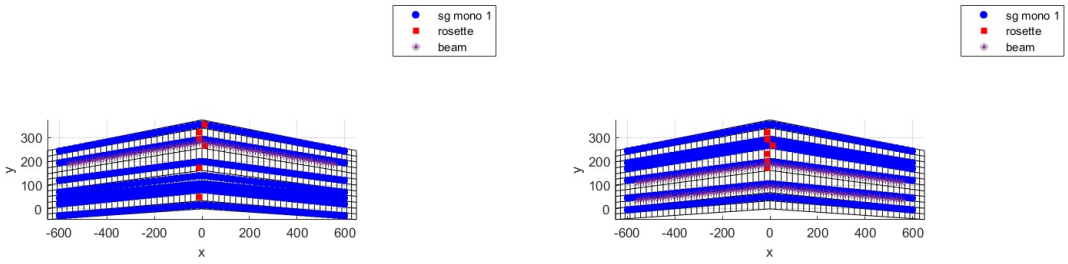

Figura C.36: Run 6 con 28 individui: PSO e WOA

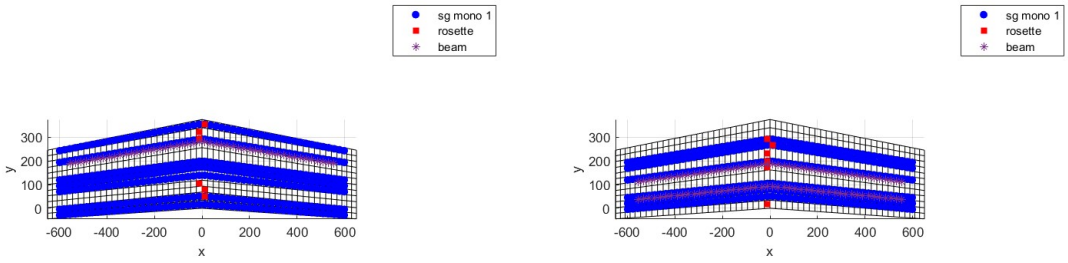

Figura C.37: Run 7 con 28 individui: PSO e WOA

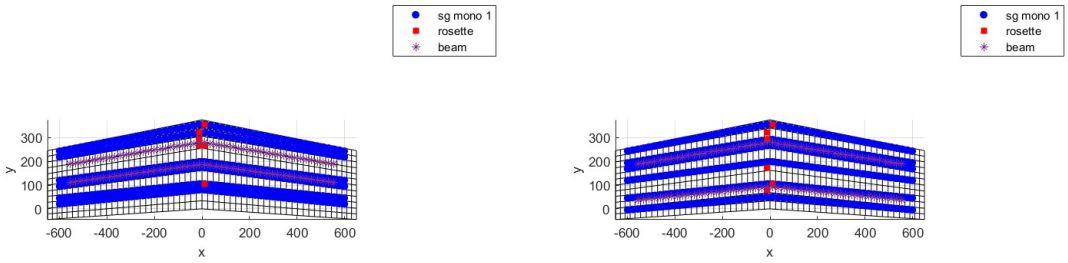

Figura C.38: Run 8 con 28 individui: PSO e WOA

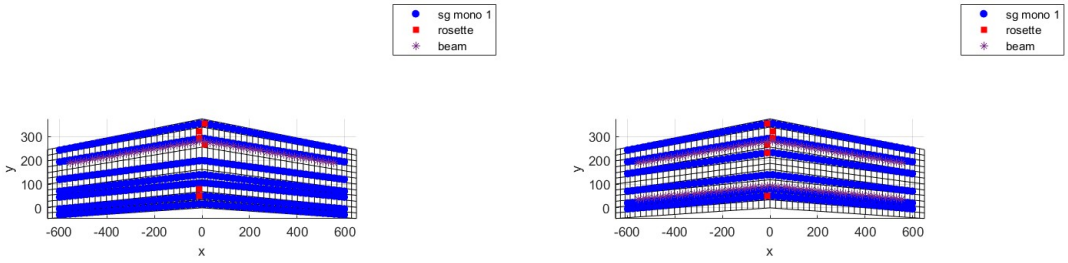

Figura C.39: Run 9 con 28 individui: PSO e WOA

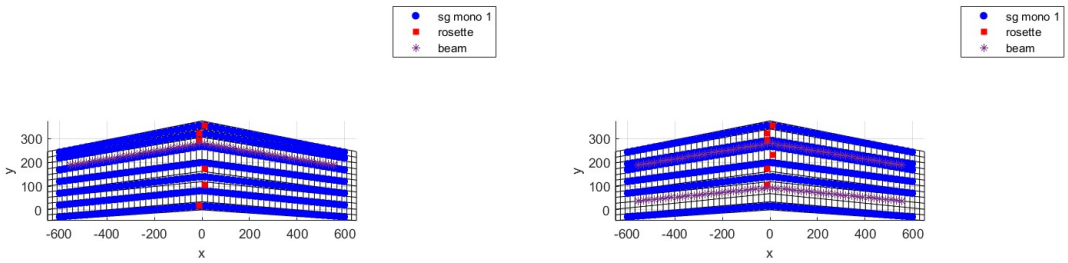

Figura C.40: Run 10 con 28 individui: PSO e WOA

Appendice D

Risultati ottimizzazione per la disposizione delle rosette

Vengono di seguito riportate le dieci ottimizzazioni effettuate in termini di andamento della miglior fitness individuata in funzione del numero di iterazioni.

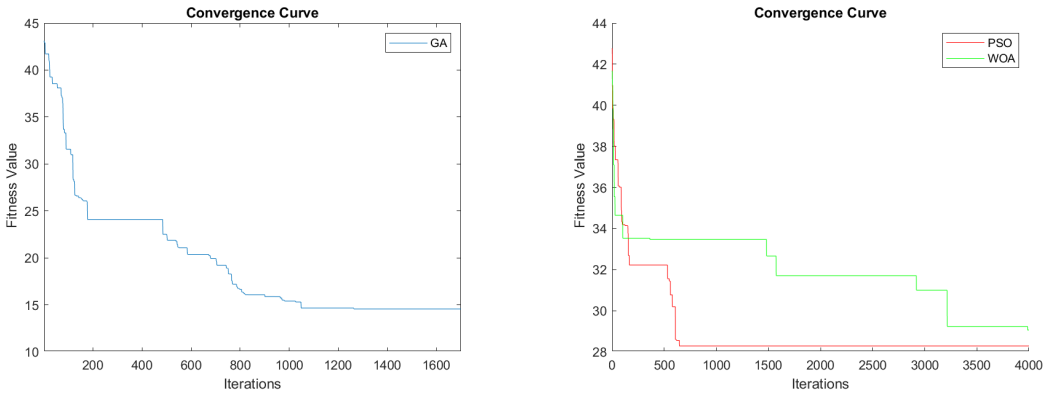

Figura D.1: Run 1: 200 individui GA e 200 individui PSO/WOA

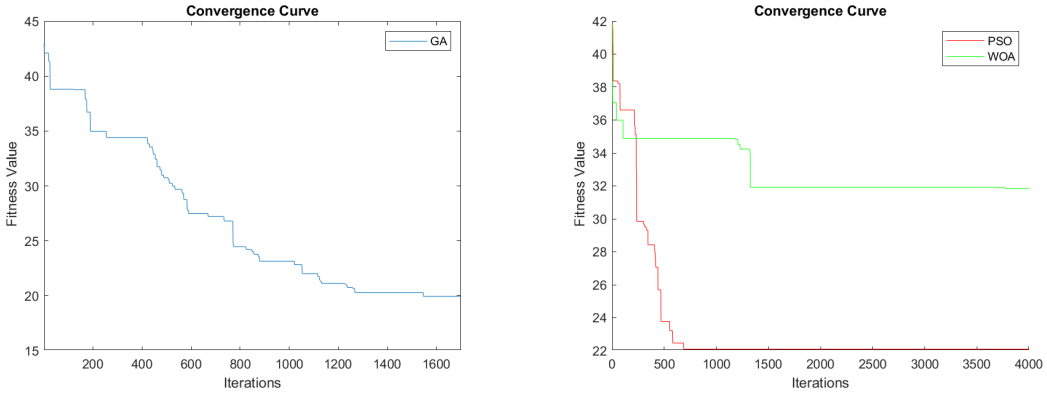

Figura D.2: Run 2: 200 individui GA e 200 individui PSO/WOA

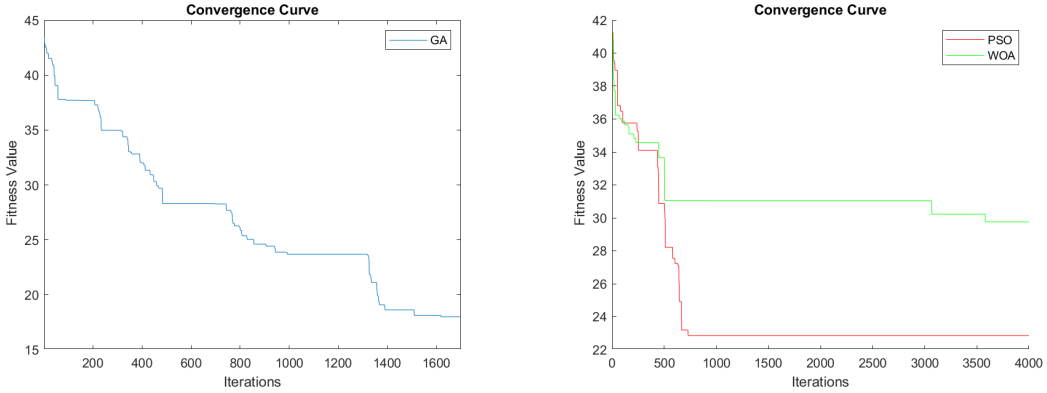

Figura D.3: Run 3: 200 individui GA e 200 individui PSO/WOA

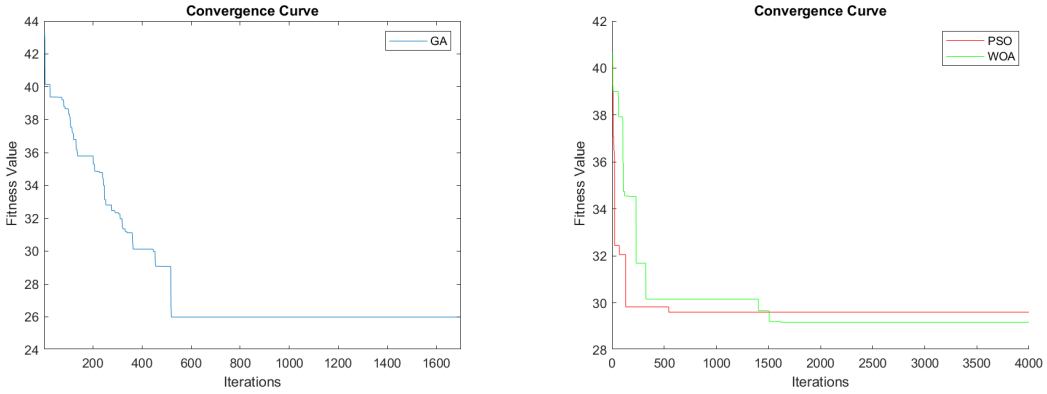

Figura D.4: Run 4: 200 individui GA e 200 individui PSO/WOA

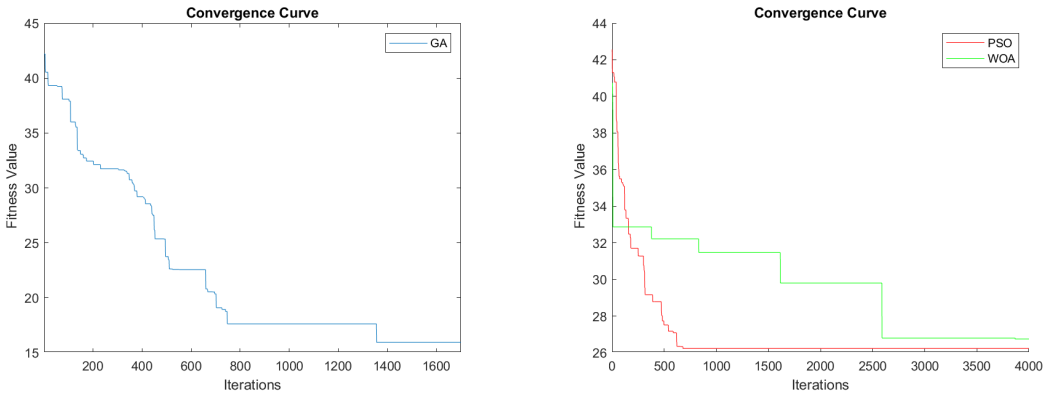

Figura D.5: Run 5: 200 individui GA e 200 individui PSO/WOA

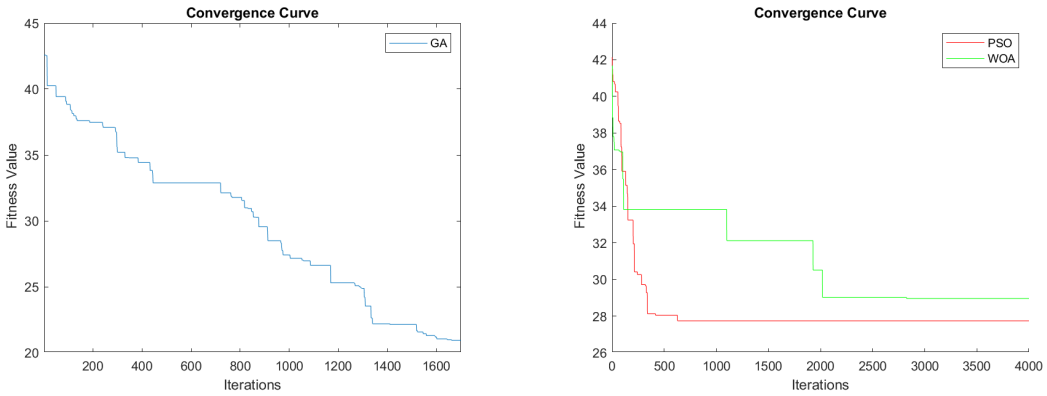

Figura D.6: Run 6: 200 individui GA e 200 individui PSO/WOA

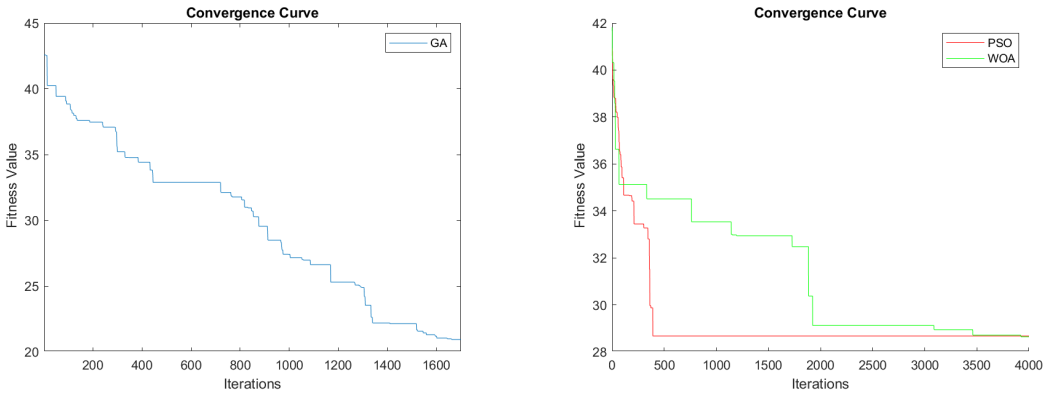

Figura D.7: Run 7: 200 individui GA e 200 individui PSO/WOA

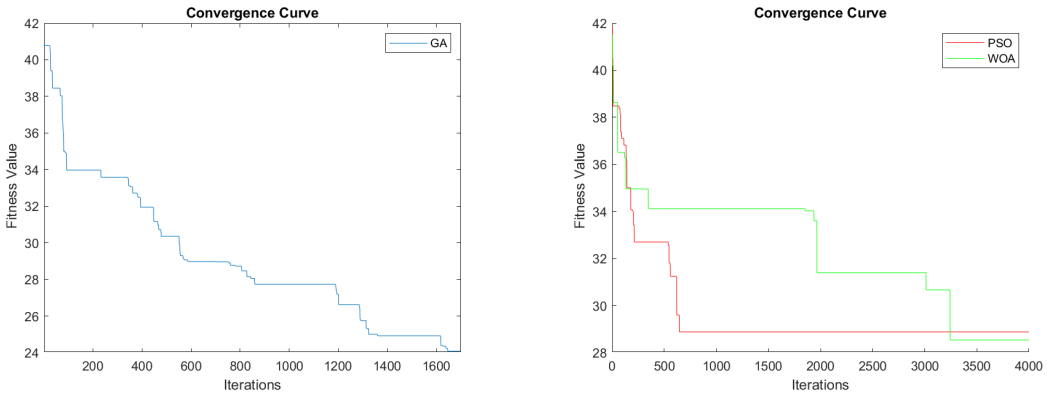

Figura D.8: Run 8: 200 individui GA e 200 individui PSO/WOA

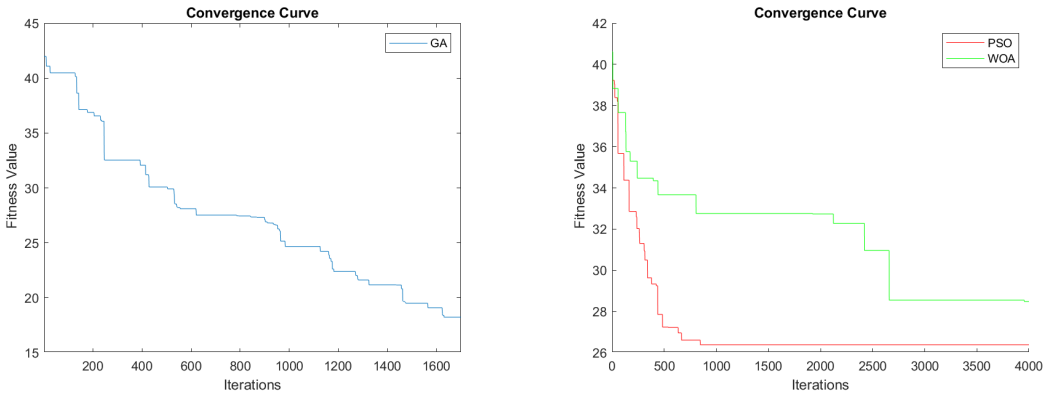

Figura D.9: Run 9: 200 individui GA e 200 individui PSO/WOA

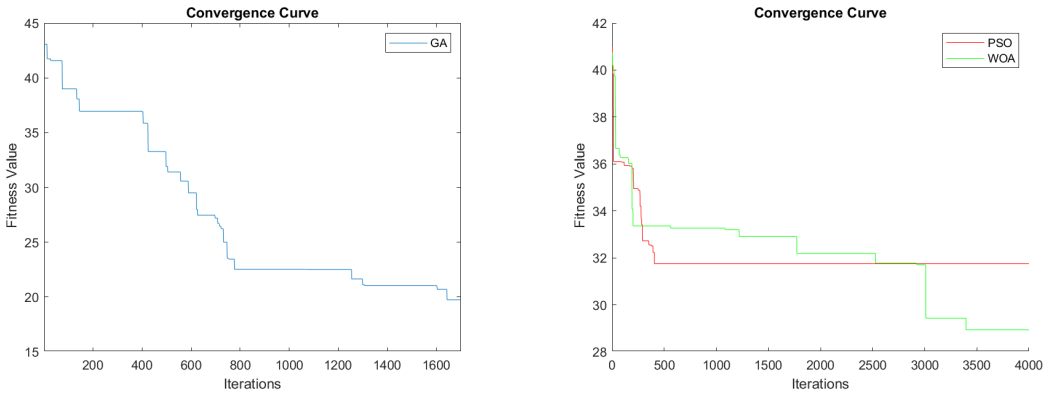

Figura D.10: Run 10: 200 individui GA e 200 individui PSO/WOA

Di seguito sono invece riportati i risultati delle 10 run in termini di best fitness, di iterazioni effettuate per individuarla e numero di valutazioni iFEM necessarie per tutti e tre gli algoritmi di ottimizzazione considerati. Inoltre sono poi riportate le configurazioni ottenute per ogni run con l'ottimizzazione eseguita con il PSO ed il WOA.

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	28.2814	646	129200
2	22.0983	688	137600
3	22.8541	729	145800
4	29.5984	544	108800
5	26.2283	680	136000
6	27.7367	628	125600
7	28.6743	391	78200
8	28.8836	646	129200
9	26.3868	846	169200
10	31.7507	406	81200

Tabella D.1: Run PSO

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	29.2335	3216	643200
2	31.9175	1327	265400
3	29.7542	3582	716400
4	29.1699	1617	323400
5	26.7443	3687	737400
6	28.956	2826	565200
7	28.6354	3923	784600
8	28.5352	3244	648800
9	28.5505	2659	531800
10	28.9401	3397	679499

Tabella D.2: Run WOA

Run	Best Fitness	Iterazione	Valutazioni iFEM
1	14.5628	1264	505600
2	19.9433	1547	618800
3	17.983	1618	647200
4	25.9841	520	208000
5	15.9406	1356	542400
6	20.9305	1662	664800
7	20.9305	1662	663800
8	24.0754	1647	658800
9	18.2354	1631	652400
10	19.7377	1644	657600

Tabella D.3: Run GA

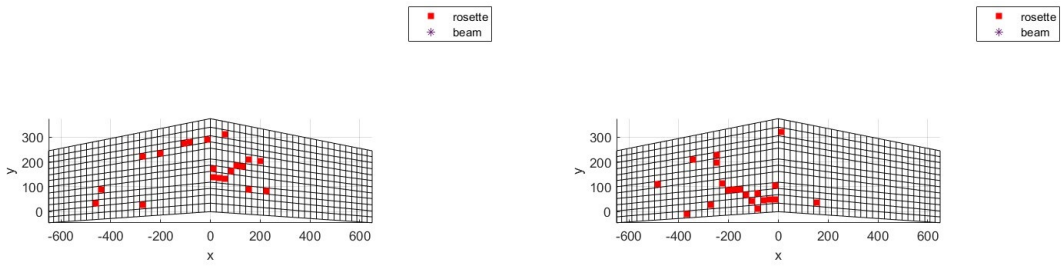

Figura D.11: Run 1 con 200 individui: PSO e WOA

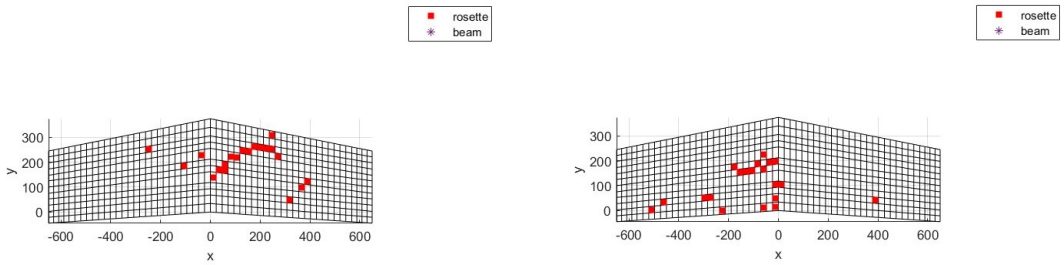

Figura D.12: Run 2 con 200 individui: PSO e WOA

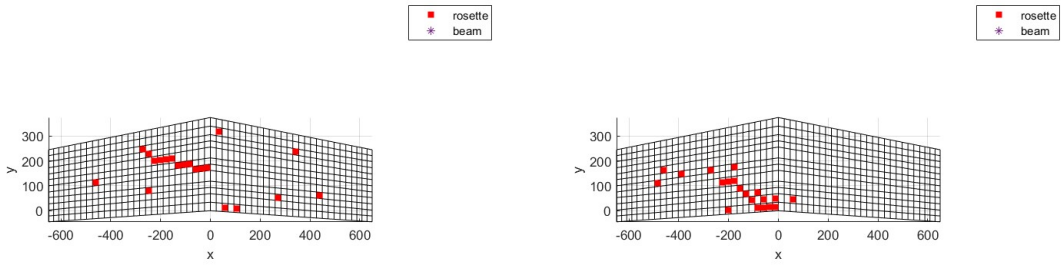

Figura D.13: Run 3 con 200 individui: PSO e WOA

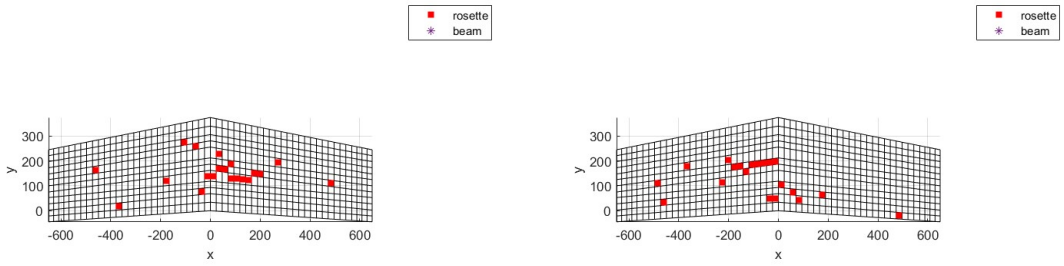

Figura D.14: Run 4 con 200 individui: PSO e WOA

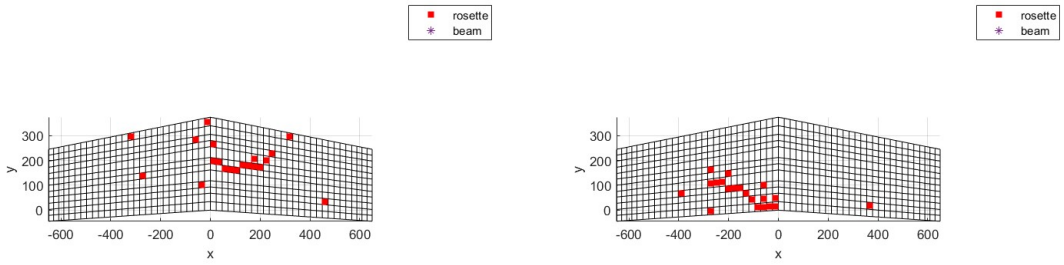

Figura D.15: Run 5 con 200 individui: PSO e WOA

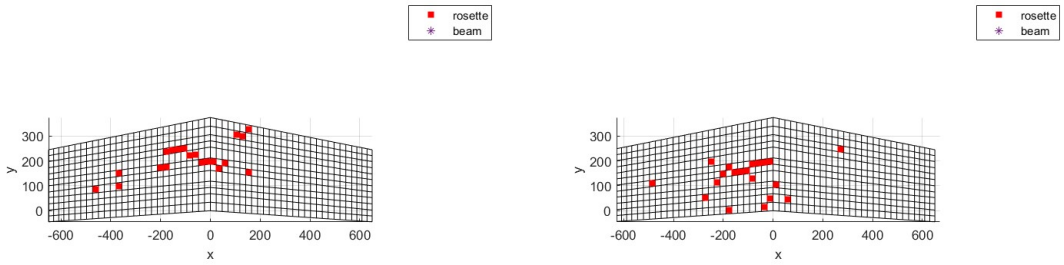

Figura D.16: Run 6 con 200 individui: PSO e WOA

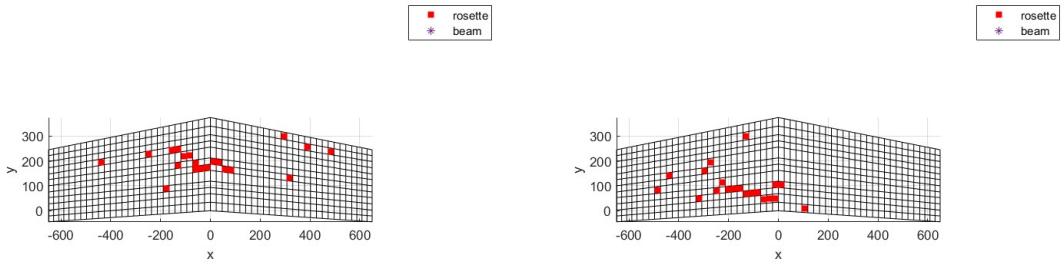

Figura D.17: Run 7 con 200 individui: PSO e WOA

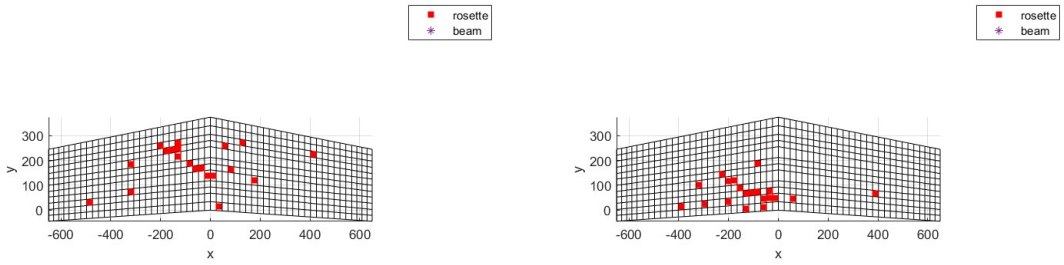

Figura D.18: Run 8 con 200 individui: PSO e WOA

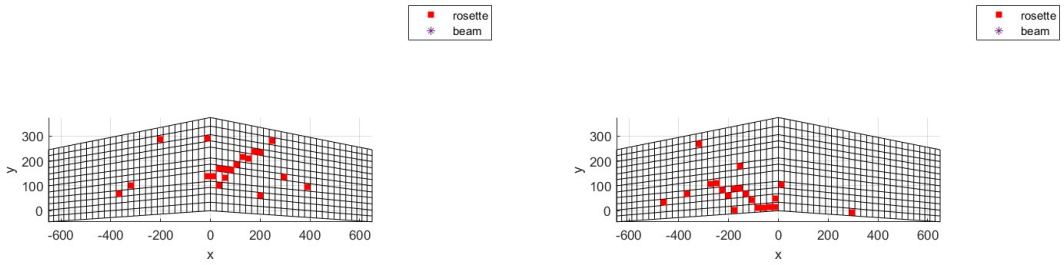

Figura D.19: Run 9 con 200 individui: PSO e WOA

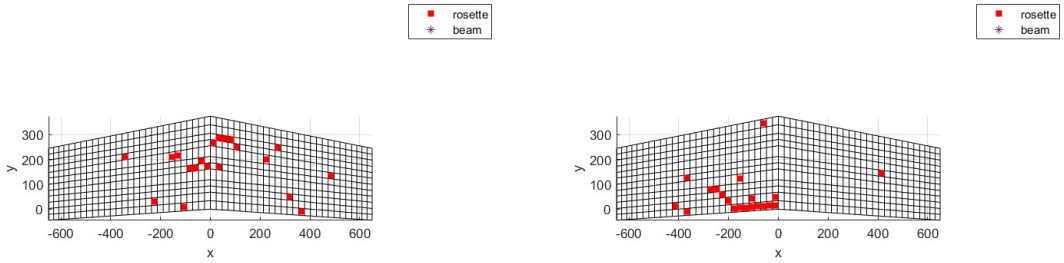

Figura D.20: Run 10 con 200 individui: PSO e WOA

Bibliografia

[1] Fernando Fausto, Adolfo Reyna-Orta, Erik Cuevas, Ángel G Andrade e Marco Perez-Cisneros. «From ants to whales: metaheuristics for all tastes». In: *Artificial Intelligence Review* 53 (2020), pp. 753–810 (cit. alle pp. 1, 3, 8).

[2] Thomas Weise. *Global optimization algorithms-theory and application*. Vol. 361. 2009 (cit. alle pp. 1, 9).

[3] Stephen Boyd, Stephen P Boyd e Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004 (cit. a p. 2).

[4] Edwin KP Chong e Stanislaw H Zak. *An introduction to optimization*. Vol. 75. John Wiley & Sons, 2013 (cit. a p. 2).

[5] Dimitri P. Bertsekas. «Nonlinear Programming». In: *Journal of the Operational Research Society* 48 (1995), p. 334 (cit. a p. 2).

[6] Arnold Neumaier. «Complete search in continuous global optimization and constraint satisfaction». In: *Acta Numerica* 13 (2004), pp. 271–369 (cit. a p. 3).

[7] Baeldung. *Deterministic and Stochastic Optimization Methods*. 2022. URL: https://www.baeldung.com/cs/deterministic-stochastic-optimization (cit. a p. 3).

[8] Johannes Schneider e Scott Kirkpatrick. *Stochastic optimization*. Springer Science & Business Media, 2007 (cit. a p. 4).

[9] James C. Spall. *Stochastic Optimization*. A cura di James E. Gentle, Wolfgang Karl Härdle e Yuichi Mori. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 173–201 (cit. a p. 4).

[10] Kenneth Sörensen, Marc Sevaux e Fred W. Glover. «A History of Metaheuristics». In: *Handbook of Heuristics*. 2015 (cit. a p. 4).

[11] Treccani. *Definizione di euristica*. URL: https://www.treccani.it/enciclopedia/euristica#:~:text=Aspetto%20del%20metodo%20scientifico%20che,del%20termine%20risalga%20a%20I. (cit. a p. 5).

[12] Tony Owen. «Heuristics: Intelligent Search Strategies for Computer Problem Solving by Judea Pearl Addison-Wesley Publishing Company, Massachusetts, USA, 101985 (£43.95)». In: *Robotica* 6.2 (1988), pp. 165–165. DOI: 10.1017/S0263574700004057 (cit. a p. 5).

[13] Andrew N. Sloss e Steven M. Gustafson. «2019 Evolutionary Algorithms Review». In: (2019) (cit. alle pp. 8, 9).

[14] Deepthi Pilakkat, S. Kanthalakshmi e S. Navaneethan. «A Comprehensive Review of Swarm Optimization Algorithms for MPPT Control of PV Systems under Partially Shaded Conditions». In: *Electronics* 24 (2020), pp. 3–14 (cit. a p. 8).

[15] Jeff Hardin Gregory Bertoni. *Il mondo della cellula*. Pearson, 2018 (cit. a p. 11).

[16] David E Goldberg. *Genetic algorithms*. pearson education India, 2013 (cit. a p. 11).

[17] Melanie Mitchell. «An introduction to genetic algorithms». In: 1996 (cit. alle pp. 11, 13).

[18] A. J. Umbarkar e P. D. Sheth. «Crossover Operators in Genetic Algorithms:A Review». In: *Soft Computing Models in Industrial and Environmental Applications*. 2015 (cit. a p. 12).

[19] Wikipedia. *Teorema degli schemi - Wikipedia, L'enciclopedia libera*. 2021. URL: https://it.wikipedia.org/wiki/Teorema_degli_schemi (cit. a p. 14).

[20] AlegsaOnline. *Teorema degli schemi - AlegsaOnline*. 2021. URL: https://it.alegsaonline.com/art/44753 (cit. a p. 14).

[21] Shi Cheng, Bin Liu, Tian Ting, Quande Qin, Yuhui Shi e Kaizhu Huang. «Survey on data science with population-based algorithms». In: *Big Data Analytics* 1 (2016), pp. 1–20 (cit. a p. 15).

[22] Andries Petrus Engelbrecht. «Computational Intelligence: An Introduction». In: 2002 (cit. alle pp. 15–17, 21).

[23] Andries Petrus Engelbrecht. «Particle swarm optimization: Velocity initialization». In: *2012 IEEE Congress on Evolutionary Computation* (2012), pp. 1–8 (cit. a p. 24).

[24] James Kennedy e Russell C. Eberhart. «A discrete binary version of the particle swarm algorithm». In: *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation* 5 (1997), 4104–4108 vol.5 (cit. a p. 26).

[25] S Mirjalili, S Mohd, G Taherzadeh, Seyedeh Zahra Mirjalili e Sohrab Salehi. «A Study of Different Transfer Functions for Binary Version of Particle Swarm Optimization». In: (apr. 2012) (cit. alle pp. 28, 29).

[26] Kodàmi. *Megattera (Megaptera novaeangliae)*. 2021. URL: https://www.kodami.it/megattera-megaptera-novaeangliae/ (cit. a p. 29).

[27] Wikipedia. *Megaptera novaeangliae*. 2021. URL: https://it.alegsaonline.com/art/44753 (cit. a p. 29).

[28] Seyedali Mirjalili e Andrew Lewis. «The whale optimization algorithm». In: *Advances in engineering software* 95 (2016), pp. 51–67 (cit. alle pp. 30, 32, 81).

[29] GC Foss e ED Haugse. «Using modal test results to develop strain to displacement transformations». In: *Proceedings of the 13th international modal analysis conference*. Vol. 2460. 1995, p. 112 (cit. a p. 36).

[30] William L Ko, W Lance Richards e Van T Tran. *Displacement theories for in-flight deformed shape predictions of aerospace structures*. Rapp. tecn. 2007 (cit. alle pp. 36, 37).

[31] A. Tessler e Jan Spangler. «A least-squares variational method for full-field reconstruction of elastic deformations in shear-deformable plates and shells». In: *Computer Methods in Applied Mechanics and Engineering* 194 (feb. 2005), pp. 327–339. DOI: 10.1016/j.cma.2004.03.015 (cit. a p. 36).

[32] Rinto Roy, Alexander Tessler, Cecilia Surace e Marco Gherlone. «Shape Sensing of Plate Structures Using the Inverse Finite Element Method: Investigation of Efficient Strain–Sensor Patterns». In: *Sensors* 20.24 (2020). ISSN: 1424-8220. DOI: 10.3390/s20247049. URL: https://www.mdpi.com/1424-8220/20/24/7049 (cit. a p. 36).

[33] Marco Gherlone, Priscilla Cerracchio, Massimiliano Mattone, Marco Di Sciuva e Alexander Tessler. «Shape sensing of 3D frame structures using an inverse Finite Element Method». In: *International Journal of Solids and Structures* 49.22 (2012), pp. 3100–3112. ISSN: 0020-7683. DOI: https://doi.org/10.1016/j.ijsolstr.2012.06.009. URL: https://www.sciencedirect.com/science/article/pii/S0020768312002648 (cit. alle pp. 39, 41).

[34] Marco Esposito. «Shape sensing and load reconstruction for aerospace structures». Tesi di dottorato. Politecnico di Torino, 2020/2021 (cit. alle pp. 48, 50).

[35] Marco Esposito e Marco Gherlone. «Composite wing box deformed-shape reconstruction based on measured strains: Optimization and comparison of existing approaches». In: *Aerospace Science and Technology* 99 (2020), p. 105758. ISSN: 1270-9638 (cit. a p. 53).

[36] Wei Chu, Xiaogang Gao e Soroosh Sorooshian. «Handling boundary constraints for particle swarm optimization in high-dimensional search space». In: *Inf. Sci.* 181 (ott. 2011), pp. 4569–4581. DOI: 10.1016/j.ins.2010.11.030 (cit. a p. 56).

[37] «Genetic Algorithms, Numerical Optimization, and Constraints». In: (mar. 2002) (cit. a p. 58).

[38] Mohammad H. Nadimi-Shahraki, Hoda Zamani e Seyedali Mirjalili. «Enhanced whale optimization algorithm for medical feature selection: A COVID-19 case study». In: *Computers in Biology and Medicine* 148 (lug. 2022), p. 105858. DOI: 10.1016/j.compbiomed.2022.105858 (cit. a p. 63).

[39] Luna Innovations. *High-Definition Continuous Fiber Grating (CFG) Sensors*. URL: https://lunainc.com/sites/default/files/assets/files/data-sheets/HD%20CFG%20Sensors%20-%20Data%20Sheet.pdf (cit. a p. 74).

[40] Luca Palmieri e Luca Schenato. «Distributed Optical Fiber Sensing Based on Rayleigh Scattering». In: *The Open Optics Journal* 7 (dic. 2013), p. 104. DOI: 10.2174/1874328501307010104 (cit. a p. 75).

[41] Scott Fitzgerald. «High Precision Fiber Optic Measurement Techniques: Luna's Optic Backscatter Reflectometer». In: (ago. 2022) (cit. a p. 75).