UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

ALGORITMI EURISTICI PER LA TARATURA DI CONTROLLORI: INSEGUIMENTO DI TRAIETTORIA IN DRONI AUTONOMI

Heuristic algorithms for controller tuning: trajectory tracking in autonomous drones

Relatore:
PROF. FREDDI ALESSANDRO

Candidata:
VECCHI CRISTINA

Correlatore:
DOTT. FELICETTI RICCARDO

A.A. 2020/2021

A mia madre, a E.e a M.C.,
senza le quali io non sarei qui.

Sommario

1 Introduzione 1

2 Gli algoritmi meta-euristici 6

2.1 Artificial Bee Colony 9

2.1.1 Le fasi dell'algoritmo 9

2.1.2 Fase di inizializzazione 11

2.1.3 Employed Bees Phase 11

2.1.4 Onlooker Bees Phase 12

2.1.5 Scout Bees Phase 12

2.2 Particle Swarm Optimization 13

2.3 Grey Wolf Optimizer 16

2.3.1 Gerarchia sociale 19

2.3.2 Accerchiamento la preda 19

2.3.3 Caccia 20

2.3.4 Attacco della preda (sfruttamento) 21

2.3.5 Ricerca di prede (esplorazione) 22

2.4 Fruit Fly Optimization Algorithm 24

2.5 Colliding Bodies Optimization 25

2.5.1 La collisione tra corpi 26

2.5.2 L'algoritmo CBO 27

2.4.1 Il coefficiente di restituzione (COR) 31

2.6 Algoritmi utilizzati 32

3 Caso di studio: L'esarotore 33

3.1 Il Modello 33

3.1.1 Cinematica e dinamica 33

3.1.2 Inclinazione del rotore 35

3.2 Leggi di controllo 37

3.2.1 Progettazione della legge di controllo Inner Loop 37
3.2.2 Progettazione della legge di controllo Outer loop 38
3.3 Schema di controllo 39
3.4 Traiettoria 40
3.5 Indici 41

4 Risultati e Confronto tra algoritmi 43
4.1 Test Preliminari 43
4.1.1 Funzione Sphere 43
4.1.2 Funzione di DeJong di quinto grado 44
4.1.3 Confronto su funzioni Sphere e DeJong 45
4.2 Applicazione degli algoritmi al drone esarotore 47
4.3 Particle Swarm Optimization modificato 52
4.3.1 Prima Simulazione del PSO logaritmico 54
4.3.2 Seconda Simulazione del PSO logaritmico 55

5 Conclusioni e sviluppi futuri 58

Bibliografia 60

Elenco delle immagini 62

Capitolo 1

Introduzione

Un “aeromobile a pilotaggio remoto” (APR), comunemente detto drone, è un velivolo caratterizzato dall’assenza del pilota a bordo, il cui volo è controllato dal computer a bordo del mezzo aereo oppure tramite il controllo remoto di un navigatore o un pilota a terra [1].

La prima “bozza” di drone è stata un’idea di Leonardo Da Vinci, nel periodo rinascimentale, che ebbe l’intuizione di una macchina in grado di volare con un decollo verticale. Da Vinci sognò e progettò una macchina che potesse rendere le persone così leggere e potenti da poter ignorare le nuvole, e ha realizzato i primi prototipi di deltaplani ed elicotteri studiando l'anatomia di umani e uccelli.

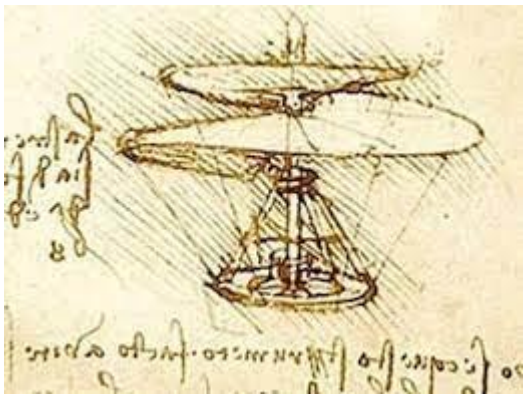


Fig. 1: Il prototipo di drone ideato da Leonardo Da Vinci

Come la maggior parte delle tecnologie e innovazioni, anche i droni sono nati per scopi bellici o militari, che includevano operazioni di ricognizione e sorveglianza che potevano essere pericolose per un pilota umano.

Il primo prototipo multi-rotore fu costruito nel 1907: il Gyroplane. Esso aveva, tuttavia, grandi limitazioni, infatti era ingombrante e richiedeva ben quattro persone per fermarlo. Durante il suo primo volo, si alzò solo sessanta centimetri da terra.

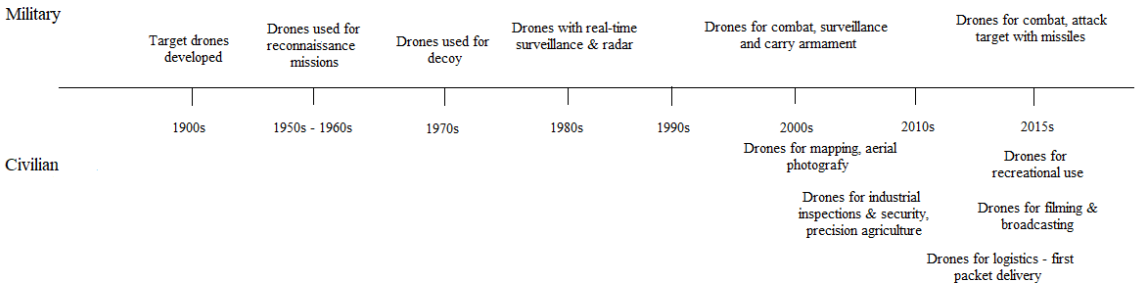


Fig. 2: Timeline dell'uso militare e civile dei droni

Nel corso del tempo, come mostra la Figura 2, tra la Prima e la Seconda Guerra Mondiale, ci fu un vero e proprio boom nello sviluppo dei droni. Lo sviluppo tecnologico consentì di concentrarsi sulla conversione di un modello di aeromobile specifico in un sistema aeromobile telecomandato che può essere controllato da un pilota automatico.

Bisognerà attendere il 2001 per avere il primo drone che effettui una missione, infatti, dopo gli eventi dell'11 settembre, la CIA ha inviato alcuni droni armati in perlustrazione in Afghanistan in cerca dei talebani. La prima operazione mirata ha avuto luogo nel febbraio 2002, quando un APR senza equipaggio è stato utilizzato per individuare un uomo sospettato di essere Osama Bin Laden. La missione è fallita: l'uomo era solo un civile, e qui sorsero i primi dubbi sull'uso di droni e intelligenza artificiale nelle azioni militari [2].

Ma con il passare dei decenni, i droni cominciarono, sempre di più, ad essere utilizzati anche per uso civile, che prevede sia l'utilizzo professionale che amatoriale o ludico, anche grazie ai sensori di cui possono essere dotati (GPS, videocamera, sensore di temperatura, ..) e alle loro dimensioni sempre più ridotte.

Protezione civile, ricerca e soccorso

Possono essere utilizzati per operazioni di ricerca e soccorso del personale, attività di monitoraggio del territorio, ispezioni di frane, perizie di esperti su aree colpite da disastri naturali e ispezioni interne ed esterne di edifici.

Un drone dà la possibilità di prendere decisioni in pochi istanti riguardanti la tipologia e la quantità di risorse che devono essere impiegate per un intervento. Se dotati di termocamere a infrarossi, gli APR permettono agli operatori di primo soccorso di individuare fonti di calore, che segnalano la presenza di persone o focolai di incendio. In questi casi, è sempre presente una sala operativa nella quale è possibile visionare le immagini catturate dal drone che consentono di prendere decisioni informate per poi spedire le squadre di soccorso sul posto individuato.

Gli APR consentono di tenere sotto controllo il lavoro svolto dalle squadre di soccorso in modo tale che essi si muovano, in tutta sicurezza, verso la posizione individuata per poi decidere se inviare rinforzi. Permettono, inoltre, l'efficace gestione delle emergenze fornendo la disponibilità di una visione chiara e completa dell'area in cui inviare i soccorsi, in modo date da prendere prontamente decisioni rapide che garantiscano la sicurezza degli operatori che salvano altre vite sfidando i pericoli.

Il multirotore esegue una rapida mappatura per facilitare l'esecuzione dell'operazione e la fase di post-emergenza. Sono utilizzati per intervenire in aree accidentate e sono solitamente dotati di due telecamere, una è una telecamera di visione e l'altra è una termocamera. La telecamera visiva, in tempo reale, permette di osservare la situazione con lo scopo di monitorare facilmente le operazioni in loco e le attrezzature disponibili. La termocamera permette di rilevare fonti di calore emesse dal corpo umano o, ad esempio, un incendio.

Poiché i droni sono in grado di volare ad una quota più bassa rispetto agli elicotteri, essi forniscono immagini più dettagliate della situazione e consentono di esplorare spazi chiusi o aree ad alto rischio in cui i veicoli dotati di pilota non possono avanzare. Il loro sistema termografico avanzato è in grado di localizzare, in breve tempo, la posizione del disastro e individuare la presenza di persone intrappolate in pochi secondi, anche nelle aree con il fumo più denso [3].

Per le spedizioni e consegne

Il gigante dell'e-commerce Amazon ha presentato il 5 giugno 2019 un drone in grado di decollare e atterrare verticalmente, che è in grado di consegnare pacchi del peso di 2.5 kg

in un raggio di consegna di circa 15 km in 30 minuti, sfruttando l'intelligenza artificiale e telecamere e sensori ottici per evitare gli ostacoli sia statici che in movimento [4].

Per esplorazioni e ricostruzioni 3D e riprese video in territori pericolosi

Un APR può essere utilizzato per realizzare rilievi topografici e fotogrammetrici che permettono di ottenere in tempi rapidi la rappresentazione tridimensionale e video di territori pericolosi, impervi e difficili da raggiungere. Inoltre, rispetto agli elicotteri, i droni possono volare anche in luoghi inaccessibili, realizzando scatti molto più vicini degli oggetti, dei terreni e dei vari contesti che si desidera immortalare [5]. Ma tali strumenti vengono utilizzati anche da ingegneri, archeologi e geologi.

Come ambulanza, ovvero dotato di defibrillatore per il primo soccorso

Il drone-ambulanza è in grado di viaggiare oltre i 100 km/h, raggiungendo la propria destinazione in pochissimo tempo, attraverso una caratteristica davvero eccezionale in termini di efficacia e velocità dei soccorsi ovvero può battere il traffico aumentando quindi le possibilità di sopravvivenza dall'8% all'80% [6].

A scopo ludico per foto e videoriprese

Se dotati di videocamere e fotocamere digitali, gli APR vengono utilizzati per realizzare riprese dall'alto di eventi sportivi di qualsiasi disciplina sportiva o semplicemente a scopo ricreativo.

Da ciò che è stato esposto, si può concludere che tali veicoli permettono la realizzazione di compiti e mansioni che possono risultare pericolosi, sporchi o noiosi, con costi notevolmente contenuti viste le alternative aeree. Quindi è molto importante, ed utile, poter realizzare il controllo del volo di questi velivoli e l'obiettivo della presente tesi si colloca in quest'ottica.

Le performance dei droni sono dettate da numerosi aspetti, sia hardware che software.
Tra gli aspetti hardware, i più rilevanti sono il peso e l'inerzia complessivi, il numero, la posizione e il dimensionamento dei motori, l'efficienza aerodinamica di pale e frame, il tipo di batteria e il tipo di sensori equipaggiati. Una modifica di tali parametri ha dei costi e richiede una nuova progettazione del drone.

Al contrario, una modifica a costo zero è rappresentata dalla modifica dei parametri software. In particolare, la taratura dei parametri di controllo ha un'elevata incidenza sulle performance di tracking di una traiettoria: ad una taratura errata può conseguire un fallimento del volo con un conseguente schianto.
Sebbene la modifica dei parametri possa essere effettuata in pochi istanti, la ricerca di un insieme di parametri soddisfacente si rivela onerosa in termini di tempo e rischiosa per l'integrità del mezzo e la sicurezza delle persone. Questa tesi, quindi, si pone come obiettivo la definizione di un metodo di taratura efficace dei parametri di controllo. Dato un modello realistico di un drone esarotore, il problema di taratura è riscritto come problema di ottimizzazione, ossia come minimizzazione di un funzionale ricercando in un determinato spazio delle soluzioni. Nella tesi si valutano diversi algoritmi di ottimizzazione meta-euristica basati su Swarm Intelligence per affrontare problemi di ottimizzazione, e si propone un algoritmo migliorato per una taratura efficace dei parametri del controllore.

La struttura di questo elaborato è definita come segue. Nel capitolo 2, dopo una breve descrizione riguardante gli algoritmi meta-euristici, verranno elencati gli aspetti più rilevanti dei paradigmi di ottimizzazione impiegati per lo svolgimento di questa attività.
Nel capitolo 3, verrà fornita una panoramica dell'esarotore preso in esame, esponendo le sue caratteristiche, il modello, la legge e lo schema di controllo, la traiettoria desiderata ed infine gli indici utilizzati.
Il capitolo 4 riguarderà il confronto tra gli algoritmi meta-euristici, prima su funzioni di benchmark in modo tale da poter comprendere il loro funzionamento, e successivamente si realizzerà un confronto più ampio tra paradigmi, che permetterà di individuare quale tra questi sia il più adatto allo scopo finale, descritto in precedenza. Infine, conclusioni ed eventuali lavori futuri sono discussi nel capitolo 5.

Capitolo 2

Gli Algoritmi Meta-Euristici

Gli algoritmi meta-euristici sono paradigmi di intelligenza computazionale particolarmente utilizzati per la risoluzione di sofisticati problemi di ottimizzazione, che permettono di approfondire la ricerca della soluzione ottima nelle zone più "promettenti" dello spazio delle soluzioni.

Ricordiamo che un problema di ottimizzazione consiste nel massimizzare o minimizzare una funzione relativa a un insieme, che rappresenta una gamma di scelte disponibili in una certa situazione. La funzione consente il confronto delle diverse scelte per determinare quale potrebbe essere la migliore [7]. Considerando un problema di minimi, si ha:

$$\min_{x} \quad f(x)$$
$$g_i(x) \leq 0,\ i = 1, \dots, m$$
$$h_i(x) = 0,\ i = 1, \dots, p$$

Dove:

$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ è la funzione obiettivo, in questo caso da minimizzare
$g_i(x) \leq 0$, chiamati vincoli di disuguaglianza
$h_i(x) = 0$, chiamati vincoli di uguaglianza

Un primo vantaggio di tali algoritmi è la loro applicabilità a un'ampia gamma di problemi ed è per questo che sono diventati così comuni e la ragione può essere riassunta in 4 caratteristiche principali [8].

- Semplicità: Sono stati per lo più ispirati da concetti molto semplici. Le ispirazioni sono tipicamente legate a fenomeni fisici, comportamenti degli animali o concetti evolutivi.

- Flessibilità: Si riferisce all'applicabilità della meta-euristica a diversi problemi senza particolari cambiamenti nella struttura dell'algoritmo, poiché assumono per lo più problemi come scatole nere, ovvero sono importanti solo gli input e gli output di un sistema. Quindi, tutto ciò di cui un designer ha bisogno è sapere come rappresentare il suo problema per la meta-euristica.
- Meccanismi privi di derivazioni: ottimizza i problemi stocasticamente, contrariamente agli approcci di ottimizzazione basati sul gradiente. Il processo di ottimizzazione inizia con una o più soluzioni casuali e non è necessario calcolare derivate per trovare l'ottimo. Ciò le rende altamente adatta a problemi reali con derivate computazionalmente costose o funzioni non differenziabili.
- Evitamento degli ottimi locali: le metaeuristiche hanno capacità superiori di evitare gli ottimi locali rispetto alle tecniche di ottimizzazione convenzionali. Ciò è dovuto alla natura stocastica delle meta-euristiche che consentono loro di evitare la stagnazione nelle soluzioni locali e di cercare estensivamente l'intero spazio di ricerca.

Come visto in [8], le meta-euristiche possono essere suddivise in due categorie principali.

- Basate su un'unica soluzione: in cui il processo di ricerca inizia con una soluzione candidata. Questa singola soluzione candidata viene quindi migliorata nel corso delle iterazioni.
- Basate sulla popolazione: eseguono l'ottimizzazione utilizzando un insieme di soluzioni (popolazione). In questo caso il processo di ricerca inizia con una popolazione iniziale casuale (soluzioni multiple), e questa popolazione viene migliorata nel corso delle iterazioni.

Le metaeuristiche basate sulla popolazione presentano alcuni vantaggi rispetto agli algoritmi basati su soluzioni singole, come detto in [8]:

- Molteplici soluzioni candidate condividono informazioni sullo spazio di ricerca, il che si traduce in salti improvvisi verso la parte promettente dello spazio di ricerca.
- Più soluzioni candidate si aiutano a vicenda per evitare soluzioni ottimali a livello locale.
- Le metaeuristiche basate sulla popolazione generalmente hanno una maggiore esplorazione rispetto agli algoritmi basati su soluzioni singole.

Gli algoritmi meta-euristici posso essere inoltre classificati in base alla natura del fenomeno simulato, e quindi si hanno [9]:

- Algoritmi evolutivi (EA), il più popolare è l'Algoritmo Genetico (Genetic Algorithm - GA). Il GA tenta di simulare il fenomeno dell'evoluzione naturale.
- Basati sulle leggi della fisica, come il Colliding Bodies Optimization (CBO)
- Swarm Intelligence (SI), come l'Artificial Bee Colony (ABC), il Grey Wolf Optimizer (GWO), il Particle Swarm Optimization (PSO) e il Fruit Fly Optimization (FOA). La SI è un ramo di ricerca che modella la popolazione di agenti interagenti o sciami in grado di auto-organizzarsi (come una colonia di formiche, uno stormo di uccelli o un sistema immunitario). Il termine sciame è usato in modo generale per riferirsi a qualsiasi raccolta contenuta di agenti o individui interagenti.

In particolare, la presente tesi si concentra sulla Swarm Intelligence (SI) che presenta i seguenti vantaggi, come descritto in [8]:

- Gli algoritmi SI conservano le informazioni sullo spazio di ricerca nel corso dell'iterazione, mentre gli algoritmi evolutivi scartano le informazioni delle generazioni precedenti.
- Gli algoritmi SI utilizzano spesso la memoria per salvare la migliore soluzione ottenuta finora.
- Gli algoritmi SI di solito hanno meno parametri da regolare.
- Gli algoritmi SI hanno meno operatori rispetto agli approcci evolutivi (crossover, mutazione, elitarismo e così via).
- Gli algoritmi SI sono facili da implementare.

Indipendentemente dalle differenze tra le meta-euristiche, una caratteristica comune è la divisione del processo di ricerca in due fasi, come definito in [8]:

- Esplorazione: si riferisce al processo di indagine delle aree promettenti dello spazio di ricerca nel modo più ampio possibile.
- Sfruttamento: si riferisce alla capacità di ricerca locale intorno alle regioni promettenti ottenute nella fase di esplorazione.

Trovare un giusto equilibrio tra queste due fasi è considerato un compito impegnativo a causa della natura stocastica della meta-euristica.

Di seguito saranno elencati e descritti gli algoritmi di ottimizzazione che sono stati studiati e impiegati per questo lavoro di tesi.

2.1 Artificial Bee Colony

L'ABC, come illustrato in [9] è un algoritmo meta-euristico population-based, che ricade nella categoria della SI che si rifà al comportamento intelligente dello sciame di api mellifere.

Nella trattazione dell'algoritmo ABC, lo sciame di api artificiali è suddiviso in tre tipologie: api operaie (employed bees), api spettatrici (onlooker bees) e api esploratrici (scout bees).
Tipicamente, la prima metà della colonia è costituita da api operaie e la restante metà è costituita dalle api spettatrici, e per ogni fonte di cibo c'è una sola ape operaia.
In altre parole, il numero di api operaie è uguale al numero di fonti di cibo intorno all'alveare. L'ape operaia la cui fonte di cibo è esaurita dalle api operaie e spettatrici, diventa un'ape scout.

La posizione di una fonte di cibo rappresenta una possibile soluzione del problema di ottimizzazione e la quantità di nettare di una fonte di cibo corrisponde alla qualità (fitness value) della soluzione associata.

2.1.1 Le fasi dell'algoritmo

Le fasi dell'algoritmo ABC sono le seguenti.

1. Inizializzazione casuale della popolazione.
2. Si ripetono i seguenti step:
   - Fase delle api operaie
   - Fase delle api spettatrici
   - Fase delle api scout

   finchè non sono soddisfatti i requisiti prestabiliti

Dopo l'inizializzazione, la popolazione delle posizioni (soluzioni) viene sottoposta a cicli ripetuti ($C = 1, 2, .., C_{max}$) del processo di ricerca delle api operaie, delle api spettatrici

e delle api scout. Ad ogni ciclo, al massimo una scout esce alla ricerca di una nuova fonte di cibo e il numero di api operaie e di api spettatrici deve rimanere sempre uguale.
La produzione di una nuova posizione di fonte di cibo si basa anche su un processo di confronto delle posizioni di fonte di cibo.
Tuttavia, le api artificiali non utilizzano alcuna informazione a confronto, ma selezionano casualmente una posizione di fonte di cibo partendo da quella esistente nella loro memoria.
Se la quantità di nettare della nuova fonte è superiore a quella della fonte precedente, l'ape memorizza la nuova posizione e dimentica quella vecchia, in caso contrario, mantiene la posizione della precedente.

Dopo che tutte le api operaie hanno completato il processo di ricerca, esse condividono le informazioni riguardo al nettare delle fonti di cibo (soluzioni) e le informazioni sulla loro posizione con le api spettatrici.

Un'ape spettatrice valuta le informazioni sul nettare e sceglie una fonte di cibo con una probabilità correlata alla sua quantità di nettare. Come nel caso dell'ape operaia, essa produce una modifica sulla posizione (soluzione) nella sua memoria e controlla la quantità di nettare della fonte candidata.

La fonte di cibo il cui nettare viene abbandonato dalle api viene sostituita con una nuova fonte di cibo dagli scout. Nell'algoritmo ABC questo viene simulato producendo casualmente una posizione e sostituendola con quella abbandonata. Ogni volta che una posizione non può essere ulteriormente migliorata attraverso un numero predeterminato di cicli chiamato limite, si abbandona quella fonte di cibo.

Dopo che ogni posizione sorgente candidata v_{ij} è stata prodotta e quindi valutata dall'ape artificiale, la sua prestazione è confrontata con quella di x_{ij}. Se il nuovo alimento ha un nettare uguale o migliore della vecchia fonte, viene sostituito con quello vecchio nella memoria. In caso contrario, il vecchio viene mantenuto. In altre parole, un meccanismo di selezione greedy viene impiegato come operazione di selezione tra le vecchie e le attuali fonti di cibo.
Inoltre, nell'algoritmo ABC, mentre le spettatrici e le api operaie svolgono il processo di sfruttamento nello spazio di ricerca, le api scout gestiscono il processo di esplorazione.

2.1.2 Fase di inizializzazione

Sono inizializzati i vettori della popolazione e vengono impostati i parametri di controllo. Poiché ogni fonte di cibo è un vettore soluzione del problema di ottimizzazione, ciascun vettore possiede n variabili che devono essere ottimizzate in modo da minimizzare la funzione obiettivo.

Per poter inizializzare la popolazione, si utilizza la seguente formula:

$$x_{ij} = x_{minj} + rand[0,1](x_{maxj} - x_{minj})$$

Dove x_{maxj} e x_{minj} sono, rispettivamente, l'upper bound e lower bound dei parametri e $rand[0,1]$ è una funzione che fornisce un numero casuale equamente distribuito nell'intervallo $[0,1]$.

2.1.3 Employed Bees Phase

Le api impiegate cercano, nell'intorno delle fonti di cibo precedentemente memorizzate, nuove fonti di cibo che possiedono più nettare. Una volta trovata, valutano la sua redditività (idoneità/fitness).
Una nuova fonte di cibo viene determinata in questo modo:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj})$$

dove x_{kj} è una fonte di cibo scelta casualmente dove i rappresenta la posizione dell'i-esima ape, mentre j rappresenta il j-esimo parametro (o j-esima dimensione) da ottimizzare. Sebbene k sia determinato casualmente, esso però deve essere diverso da i, inoltre φ_{ij} è un numero casuale compreso tra $[-1, 1]$. Tale valore controlla la produzione di una posizione di fonte di cibo vicina intorno a x_{ij} e la modifica rappresenta il confronto visivo delle posizioni di cibo vicino da parte dell'ape.
Dopo aver prodotto la nuova fonte di cibo viene calcolata la sua fitness function.

É da notare che, al diminuire della differenza tra i parametri di x_{ij} e x_{kj}, diminuisce anche la perturbazione sulla posizione x_{ij}. Pertanto, quando la ricerca si avvicina alla

soluzione ottimale nello spazio di ricerca, la lunghezza del passo viene ridotta in modo adattativo.
Se un parametro prodotto da questa operazione supera il suo limite predeterminato, il parametro deve essere forzato nel range accettabile.

2.1.4 Onlooker Bees Phase

Le api non occupate sono costituite da due gruppi di api: api spettatrici e scout. Le api operaie condividono le loro informazioni sulla fonte di cibo con le api spettatrici nell'alveare e quindi le api interessate scelgono, randomicamente, le fonti di cibo in base a queste informazioni.
Nell'ABC, un'ape spettatrice sceglie una fonte di cibo in base al valore della probabilità calcolata utilizzando i valori di fitness forniti dalle api operaie. A tale scopo, può essere utilizzata una tecnica di selezione basata sulla fitness function, come il metodo della Roulette Wheel Selection.

Il valore di probabilità P_i con il quale la fonte di cibo viene scelta da un'ape spettatrice può essere calcolato come:

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i}$$

dove fit_i è il valore di fitness della soluzione i valutato dall'ape operaia, che è proporzionale alla quantità di nettare della fonte di cibo nella posizione i e SN è il numero di fonti di cibo, che è uguale al numero di api operaie. In questo modo, le api operaie scambiano le loro informazioni con le spettatrici.

2.1.5 Scout Bees Phase

Le api che scelgono casualmente le loro fonti di cibo sono chiamate scout. Le api operaie le cui soluzioni non possono essere migliorate dopo un predeterminato numero di prove, specificato dall'utente e qui chiamato "limite" o "criterio di abbandono", diventano scout e le precedenti soluzioni vengono abbandonate. Quindi, le api scout iniziano a cercare nuove soluzioni in modo casuale. Quindi quelle fonti che sono inizialmente povere o sono state rese povere dallo sfruttamento vengono abbandonate.

2.2 Particle Swarm Optimization

Come esposto in [10], il Particle Swarm Optimization è un algoritmo meta-euristico population-based. Ricade nella categoria degli algoritmi SI e si basa sul comportamento sociale di stormi di uccelli e banchi di pesci. Con questo approccio, la soluzione ottima del problema di ottimizzazione deriva dall'interazione sociale delle particelle, in quanto solo come gruppo riusciranno ad affrontare il problema.

L'algoritmo prevede la generazione e la gestione di un insieme di soluzioni chiamato popolazione, dove ciascuna particella rappresenta una possibile soluzione candidata a cui è associato un valore (fitness value) che indica la sua qualità. Ogni particella è caratterizzata da una posizione e da una velocità.

Un numero prefissato di particelle è inizialmente posizionato in modo del tutto casuale all'interno dello spazio delle soluzioni ammissibili. Ogni particella esplora l'aera di ricerca e determina il successivo movimento in base alla propria posizione corrente, la propria migliore posizione passata (personal best fitness) e la migliore posizione passata dello sciame (global best fitness), il tutto scambiando informazioni con le altre particelle. Con questo approccio, lo sciame tenderà a dirigersi verso la migliore posizione individuata globalmente grazie all'iterazione e lavoro di gruppo.

Il problema PSO è definito da M particelle, da N variabili e dalla fitness function, $f : S \in \mathbb{R}^n \rightarrow \mathbb{R}$, che definisce la bontà della posizione raggiunta dall'i-esima particella.

Come già detto, all'iterazione k, è caratterizzata dalle seguenti variabili.

- x_i^k: attuale posizione della i-esima particella
- v_i^k: attuale velocità della i-esima particella
- p_i: posizione migliore passata della i-esima particella (personal best)
- $f(x_i^k)$: valore della fitness fuction nella posizione attuale per la i-esima particella
- $pbest_i = f(p_i)$: valore della fitness function calcolata nella migliore posizione passata dalla i-esima particella

Il $pbest_i$ è un parametro importante perché, ad ogni iterazione, la nuova migliore soluzione (posizione) viene confrontata con esso.

Gli step dell'algoritmo PSO sono i seguenti:

1. Si inizializza una popolazione di particelle con posizioni e velocità casuali nello spazio di ricerca;
2. Si ripete il seguente loop:
   a) Per ogni particella, si valuta la funzione fitness $f\left(x_i^k\right)$ nella posizione attuale per la particella i-esima;
   b) Si confronta la fitness function trovata con il $pbest_i$
   c) Se il valore corrente è migliore del $pbest_i$, si aggiorna $pbest_i$ al valore corrente
   d) Si individua la particella nell'intorno con il miglior valore della fitness function a livello globale, e le si assegna l'indice g (global)
   e) Si aggiorna la velocità e la posizione della particella secondo la seguente equazione:

$$\begin{cases} \vec{v}_i \leftarrow \vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i) \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \end{cases}$$

   f) Si itera finché non viene soddisfatto un certo criterio, tipicamente un numero massimo di iterazioni o una fitness function sufficientemente buona;
3. Fine delle iterazioni.

Dove:

- $U(0,\phi_1)$ rappresenta un vettore di numeri casuali distribuiti uniformemente in $[0,\phi_1]$ che viene generato casualmente ad ogni iterazione e per ogni particella
- $\otimes$ è la moltiplicazione per componente
- $p_i - x_i^k$ rappresenta la distanza dalla migliore posizione visitata dalla singola particella
- $p_g - x_i^k$ rappresenta la distanza dalla migliore posizione visitata da tutto lo sciame

La passata esperienza di ogni particella è definita come $U(0,\phi_1) \otimes (p_i - x_i^k)$, mentre la collaborazione tra particelle è definita come $U(0,\phi_2) \otimes (p_g - x_i^k)$.

Il PSO appena descritto ha un numero limitato di parametri che devono essere fissati, uno dei parametri è la dimensione della popolazione.

Questo è spesso impostato empiricamente sulla base della dimensionalità e della difficoltà percepita di un problema, e comunemente si utilizza una popolazione con valori compresi tra 20 e 50.
I parametri ϕ_1 e ϕ_2 determinano l'ampiezza o intensità delle forze casuali che attraggono le particelle verso la migliore posizione visitata in passato e verso la miglior posizione visitata dalle particelle nell'intorno, inoltre ϕ_1 e ϕ_2 sono spesso chiamati coefficienti (o costanti) di accelerazione.
Per evitare l'instabilità dovuta ad incrementi senza controllo della velocità delle particelle è opportuno limitare la velocità v_i nell'intervallo $[-V_{max}, +V_{max}]$. La scelta del parametro V_{max} richiede una certa attenzione poiché sembra influenzare l'equilibrio tra esplorazione e sfruttamento. Infatti, un V_{max} grande facilita l'esplorazione globale mentre un V_{max} piccolo incoraggia lo sfruttamento locale. Una possibile soluzione all'utilizzo del V_{max}, e di conseguenza avere un maggior controllo sull'esplorazione e sullo sfruttamento, è stato quello di introdurre il peso di inerzia ω, che va a moltiplicare la velocità v_i:

$$\begin{cases} \vec{v}_i \leftarrow \omega\vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i) \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \end{cases}$$

Interpretando $U(0,\phi_1) \otimes (p_i - x_i) + U(0,\phi_2) \otimes (p_g - x_i)$ come la forza esterna f_i che agisce su una particella, allora il cambiamento della velocità di una particella (cioè l'accelerazione della particella) può essere scritta come $\Delta v_i = f_i - (1-\omega)v_i$. Ovvero, la costante $(1-\omega)$ agisce come coefficiente di attrito, e quindi ω può essere interpretata come la viscosità del mezzo in cui si muove una particella.

Migliori prestazioni potrebbero essere ottenute impostando inizialmente ω su un valore relativamente alto, ad esempio 0.9, che corrisponde a un sistema in cui le particelle si muovono in un mezzo a bassa viscosità ed eseguono esplorazioni estese, per poi ridurre gradualmente (in proporzioni al numero di iterazioni) ω ad un valore molto più basso, ad esempio 0.4, per il quale il sistema sarebbe più dissipativo.
Facendo in questo modo, all'inizio si intensifica la fase di esplorazione in modo tale da poter individuare le zone più promettenti, per poi realizzare lo sfruttamento nelle aree individuate.

È anche possibile partire da valori di $\omega > 1$, che renderebbero inizialmente instabile lo sciame, a patto che il valore sia sufficientemente ridotto da portare lo sciame in una regione stabile (il valore preciso di ω che garantisce stabilità dipende dai valori dello sciame e dai coefficienti di accelerazione). Naturalmente possono essere adottate altre strategie per regolare il peso inerziale, per esempio sono stati ottenuti miglioramenti significativi delle prestazioni del attraverso l'adattamento di ω sfruttando un sistema fuzzy. Un'altra strategia efficace consiste nell'utilizzare un peso inerziale con una componente casuale, piuttosto che una riduzione del tempo.
Un modo alternativo per garantire la convergenza della particella, prevenire l'esplosione, ed eliminare il parametro arbitrario V_{max}, è l'applicazione di un coefficiente di costrizione χ sulla velocità:

$$\begin{cases} \vec{v}_i \leftarrow \chi(\vec{v}_i + \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0,\phi_2) \otimes (\vec{p}_g - \vec{x}_i)) \\ \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \end{cases}$$

Dove $\phi = \phi_1 + \phi_2 > 4$ e:

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}$$

Quando viene utilizzato il metodo di costrizione, ϕ è comunemente impostato su 4.1, $\phi_1 = \phi_2$ e il moltiplicatore costante χ è circa 0.7298. Ciò si traduce nella velocità precedente che viene moltiplicata per 0.7298 e ciascuno dei due termini $(p - x)$ viene moltiplicato per un numero casuale limitato da 0.7298 × 2,05 ≈ 1.49618. È da notare che il PSO con costrizione è del tutto equivalente al PSO con il peso di inerzia. Infatti, è sempre possibile passare dalla formula del peso di inerzia alla formula del peso di costrizione tramite la mappatura $\omega \leftrightarrow \chi$ e $\phi_i \leftrightarrow \chi\phi_i$.

2.3 Grey Wolf Optimizer

Il GWO, come detto in [8], è un algoritmo meta-euristico population-based, che ricade nella categoria degli algoritmi SI e si basa sul comportamento e la gerarchia di leadership dei lupi grigi.
Il lupo grigio (Canis lupus) appartiene alla famiglia dei Canidae e sono considerati predatori che si trovano in cima alla catena alimentare. I lupi grigi preferiscono per lo più

vivere in branco, dove generalmente la dimensione del gruppo è composto da 5-12 elementi.

Di particolare interesse è che hanno una gerarchia sociale dominante molto rigida, composta da 4 tipologie di lupi:

Fig. 3: Gerarchia dei lupi grigi

In cima alla gerarchia ci sono i lupi alfa, i leader, che sono un maschio e una femmina. L'alfa è principalmente responsabile delle decisioni che riguardano la caccia, il posto per dormire, l'ora di svegliarsi ecc. La decisione sull'elezione dell'alfa è dettata dal branco.
Tuttavia, è stato anche osservato un qualche tipo di comportamento democratico, in cui un lupo alfa segue gli altri lupi nel branco.

Nei raduni, l'intero branco riconosce l'alfa tenendo la coda abbassata. Il lupo alfa è anche chiamato lupo dominante, poiché i suoi ordini dovrebbero essere seguiti dal branco. I lupi alfa possono accoppiarsi solo nel branco. È interessante notare che l'alfa non è necessariamente il membro più forte del branco, ma il migliore in termini di gestione del branco.
Il secondo livello nella gerarchia dei lupi grigi è beta. I beta sono lupi subordinati che aiutano l'alfa nel processo decisionale o in altre attività del branco. Il lupo beta può essere maschio o femmina, ed è probabilmente il miglior candidato per sostituire l'alfa nel caso in cui uno di essi muoia o diventi troppo vecchio. Svolge il ruolo di consigliere per l'alfa e disciplinatore per il branco. Il lupo beta rafforza i comandi dell'alpha in tutto il branco e fornisce feedback all'alpha.
Il lupo grigio di rango più basso è l'omega. Sono gli ultimi lupi che possono mangiare. Può sembrare che l'omega non sia un individuo importante nel branco, ma è stato osservato che l'intero branco deve affrontare lotte interne e problemi in caso di perdita dell'omega.

Se un lupo non è un alfa, beta o omega, è chiamato subordinato (o delta in alcuni riferimenti). I lupi delta devono sottomettersi ad alfa e beta, ma dominano l'omega.
Appartengono a questa categoria scout, sentinelle, anziani, cacciatori e custodi:

- Gli scout sono responsabili della sorveglianza dei confini del territorio e dell'avvertimento del branco in caso di pericolo
- Le sentinelle proteggono e garantiscono la sicurezza del branco
- Gli anziani sono i lupi esperti che erano alfa o beta
- I cacciatori aiutano gli alfa e i beta a cacciare le prede e a fornire cibo al branco
- I custodi sono responsabili della cura dei lupi deboli, malati e feriti nel branco

Oltre alla gerarchia sociale dei lupi, la caccia di gruppo è un altro interessante comportamento sociale dei lupi grigi.
Le fasi principali della caccia al lupo grigio sono le seguenti:

- Inseguire e avvicinarsi alla preda.
- Circondare e molestare la preda finché non smette di muoversi.
- Attaccare verso la preda.

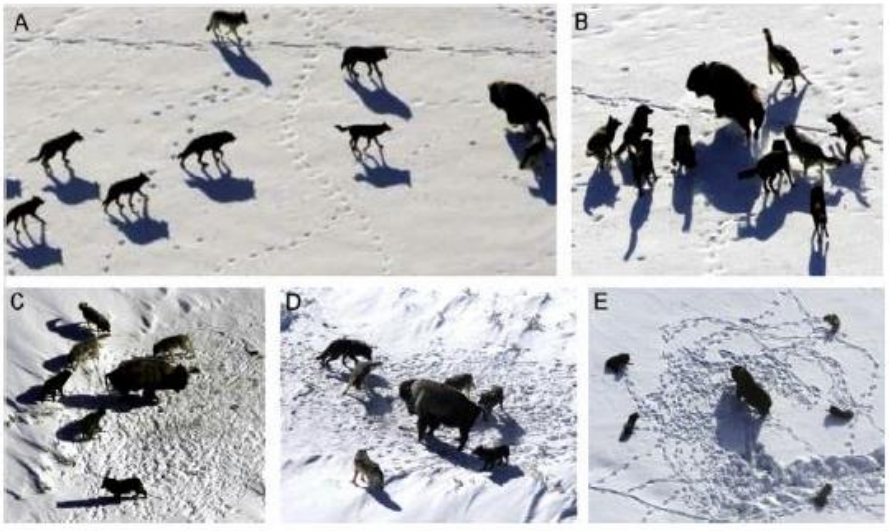


Fig. 4: Comportamento dei lupi grigi nella caccia: (A) Inseguire e avvicinarsi alla preda, (B-D) Inseguire, circondare e molestare la preda, (E) situazione stazionaria e attacco alla preda

Vengono di seguito forniti i modelli matematici della gerarchia sociale, del monitoraggio, dell'accerchiamento e dell'attacco delle prede.

2.3.1 Gerarchia sociale

Al fine di modellare matematicamente la gerarchia sociale dei lupi durante la progettazione di GWO, consideriamo la soluzione più adatta come l'alfa (α). Di conseguenza, la seconda e la terza migliore soluzione sono denominate rispettivamente beta (β) e delta (δ). Si assume che le altre soluzioni candidate siano omega (ω).
Nell'algoritmo GWO la ricerca (ottimizzazione) è guidata da α, β e δ, mentre i lupi ω seguono questi tre lupi.

2.3.2 Accerchiamento la preda

Come accennato in precedenza, i lupi grigi circondano la preda durante la caccia.
Per modellare matematicamente il comportamento di accerchiamento vengono proposte le seguenti equazioni:

$$\vec{D} = \left|\vec{C} \otimes \vec{X}_p(t) - \vec{X}(t)\right|$$

$$\vec{X}(t+1) = \left|\vec{X}_p(t) - \vec{A} \otimes \vec{D}\right|$$

Dove t indica l'iterazione corrente, $\overrightarrow{A}$ e $\vec{C}$ sono vettori di coefficienti, $\overrightarrow{\mathrm{X_p}}$ è il vettore di posizione della preda e $\vec{\mathrm{X}}$ indica il vettore di posizione di un lupo grigio.
I vettori $\overrightarrow{A}$ e $\vec{C}$ sono calcolati come segue:

$$\vec{A} = 2\vec{a} \otimes \vec{r}_1 - \vec{a}$$

$$\vec{C} = 2 \otimes \vec{r}_2$$

dove i componenti di $\vec{\mathrm{a}}$ sono diminuiti linearmente da 2 a 0 nel corso delle iterazioni e $\overrightarrow{r_1}$, $\overrightarrow{r_2}$ sono vettori casuali in $[0, 1]$.
Un esempio con un vettore posizione bidimensionale o tridimensionale e alcuni dei possibili vicini sono illustrati in Fig. 5(a).

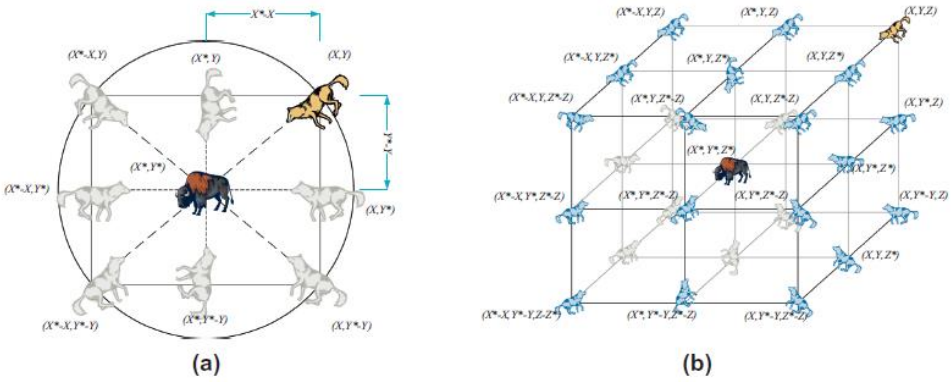


Fig. 5: Vettori posizione in 2D e 3D e le loro prossime possibili locazioni

Come si può vedere in Fig. 5, un lupo grigio nella posizione di (X, Y) può aggiornare la sua posizione in base alla posizione della preda (X^*, Y^*). È possibile raggiungere punti diversi intorno all'agente migliore rispetto alla posizione corrente regolando il valore dei vettori $\overrightarrow{A}$ e $\vec{C}$. Ad esempio, $(X^* - X, Y^*)$ può essere raggiunto ponendo $A = (1,0)$ e $C = (1,1)$. Le possibili posizioni aggiornate di un lupo grigio nello spazio 3D sono illustrate in Fig. 5(b). Si noti che i vettori casuali $\overrightarrow{r_1}$ e $\overrightarrow{r_2}$ consentono ai lupi di raggiungere qualsiasi posizione tra i punti illustrati in Fig. 5.
Quindi un lupo grigio può aggiornare la sua posizione all'interno dello spazio attorno alla preda in qualsiasi posizione casuale utilizzando le equazioni illustrate. Lo stesso concetto può essere esteso ad uno spazio di ricerca di n dimensioni, e i lupi grigi si muoveranno in ipercubi (o ipersfere) attorno alla migliore soluzione finora ottenuta.

2.3.3 Caccia

I lupi grigi hanno la capacità di riconoscere la posizione delle prede e circondarle. La caccia è solitamente guidata dall'alfa. Il beta e il delta potrebbero anche partecipare alla caccia occasionalmente. Tuttavia, in uno spazio di ricerca astratto non abbiamo idea della posizione dell'ottimo (preda).

Per simulare matematicamente il comportamento di caccia dei lupi grigi, supponiamo che l'alfa (migliore soluzione candidata) beta e delta abbiano una migliore conoscenza della potenziale posizione della preda. Pertanto, salviamo le prime tre migliori soluzioni ottenute finora e obblighiamo gli altri agenti di ricerca (compresi gli omega) ad aggiornare le loro posizioni in base alla posizione dei migliori agenti di ricerca. Si propongono al riguardo le seguenti formule:

$$\vec{D}_\alpha = |\vec{C}_1 \otimes \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \otimes \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \otimes \vec{X}_\delta - \vec{X}|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \otimes (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \otimes (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \otimes (\vec{D}_\delta)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$

La Fig. 6 mostra come un agente di ricerca aggiorna la sua posizione secondo alfa, beta e delta in uno spazio di ricerca 2D. Si può osservare che la posizione finale sarebbe in un luogo casuale all'interno di un cerchio definito dalle posizioni di alfa, beta e delta nello spazio di ricerca. In altre parole, alfa, beta e delta stimano la posizione della preda e altri lupi aggiornano le loro posizioni in modo casuale attorno alla preda.

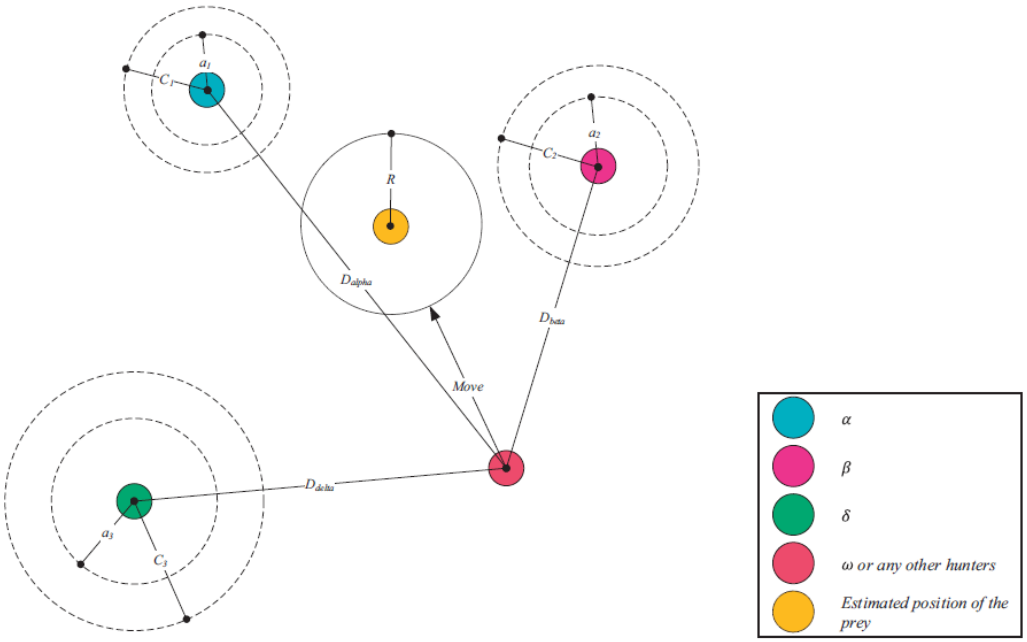


Fig. 6: Aggiornamento delle posizioni nel GWO

2.3.4 Attacco della preda (sfruttamento)

Come accennato in precedenza i lupi grigi terminano la caccia attaccando la preda quando smette di muoversi. Per modellare matematicamente l'avvicinamento alla preda diminuiamo il valore di $\vec{a}$. Notare che anche l'intervallo di fluttuazione di $\vec{A}$ è diminuito di $\vec{a}$. In altre parole, $\vec{A}$ è un valore casuale nell'intervallo $[-2a, 2a]$ dove a viene decrementato da 2 a 0 nel corso delle iterazioni. Quando i valori casuali di $\vec{A}$ sono in $[-1, 1]$, la posizione successiva di un agente di ricerca può essere in qualsiasi posizione tra la sua posizione attuale e la posizione della preda.

La Fig. 7(a) mostra che $|\vec{A}| < 1$ costringe i lupi ad attaccare verso la preda. Con gli operatori proposti finora, l'algoritmo GWO consente ai suoi agenti di ricerca di aggiornare la propria posizione in base alla posizione di alfa, beta e delta, e poi attaccare verso la preda.

Tuttavia, l'algoritmo GWO è soggetto a stagnazione nelle soluzioni locali con questi operatori. È vero che il meccanismo di accerchiamento proposto mostra l'esplorazione in una certa misura, ma GWO ha bisogno di più operatori per enfatizzare l'esplorazione.

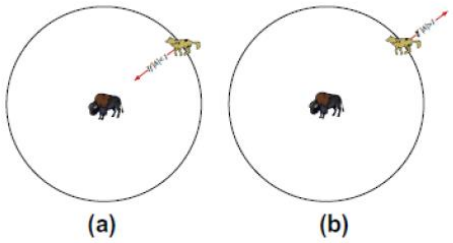


Fig. 7: Attacco della preda (a) vs Ricerca della preda (b)

2.3.5 Ricerca di prede (esplorazione)

I lupi grigi cercano principalmente in base alla posizione di alfa, beta e delta. Si discostano l'uno dall'altro per cercare la preda e convergono per attaccare la preda. Per modellare matematicamente la divergenza, utilizziamo $\vec{A}$ con valori casuali maggiori di 1 o minori di -1 per obbligare l'agente di ricerca a divergere dalla preda. Ciò enfatizza l'esplorazione e consente all'algoritmo GWO di eseguire ricerche a livello globale. La Fig. 7(b) mostra anche che $|\vec{A}| > 1$ costringe i lupi grigi a divergere dalla preda per trovare una preda più adatta.

Un'altra componente di GWO che favorisce l'esplorazione è $\vec{C}$. Il vettore $\vec{C}$ contiene valori casuali in $[0, 2]$. Questo componente fornisce pesi casuali per la preda al fine di enfatizzare stocasticamente ($\vec{C} > 1$) o deenfatizzare ($\vec{C} < 1$) l'effetto della preda nel definire la distanza.

Questo aiuta GWO a mostrare un comportamento più casuale durante l'ottimizzazione, favorendo l'esplorazione e l'evitamento dell'ottimo locale. Vale la pena ricordare qui che $\vec{C}$ non è linearmente diminuito rispetto ad $\vec{A}$.

Richiediamo deliberatamente a $\vec{C}$ di fornire valori casuali in ogni momento per enfatizzare l'esplorazione non solo durante le iterazioni iniziali ma anche nelle iterazioni

finali. Questo componente è molto utile in caso di stagnazione in ottimi locali, specialmente nelle iterazioni finali.

Il vettore $\vec{C}$ può essere considerato anche come l'effetto degli ostacoli all'avvicinamento della preda in natura. In generale, gli ostacoli in natura compaiono nei percorsi di caccia dei lupi e di fatto impediscono loro di avvicinarsi rapidamente e comodamente alla preda. Questo è esattamente ciò che fa il vettore $\vec{C}$. A seconda della posizione di un lupo, può dare casualmente un peso alla preda e rendere più difficile e più lontano raggiungere i lupi, o viceversa.

Per riassumere, il processo di ricerca inizia con la creazione di una popolazione casuale di lupi grigi (soluzioni candidate) nell'algoritmo GWO. Nel corso delle iterazioni, i lupi alfa, beta e delta stimano la probabile posizione della preda. Ogni soluzione candidata aggiorna la sua distanza dalla preda.

Il parametro $\vec{a}$ viene diminuito da 2 a 0 per enfatizzare rispettivamente l'esplorazione e lo sfruttamento. Le soluzioni candidate tendono a divergere dalla preda quando $|\vec{A}| > 1$ e convergono verso la preda quando $|\vec{A}| < 1$. Infine, l'algoritmo GWO è terminato dalla soddisfazione di un criterio finale.

Per vedere come GWO è teoricamente in grado di risolvere problemi di ottimizzazione, si possono notare alcuni punti:

- La gerarchia sociale proposta aiuta GWO a salvare le migliori soluzioni ottenute finora nel corso dell'iterazione.
- Il meccanismo di accerchiamento proposto definisce un intorno a forma di cerchio attorno alle soluzioni che può essere esteso a dimensioni superiori come un'ipersfera.
- I parametri casuali $\vec{A}$ e $\vec{C}$ aiutano le soluzioni candidate ad avere ipersfere con raggi casuali diversi.
- Il metodo di caccia proposto consente soluzioni candidate per individuare la probabile posizione della preda.
- L'esplorazione e lo sfruttamento sono garantiti dai valori adattativi di a e $\vec{A}$.
- I valori adattativi dei parametri *a* e *A* consentono a GWO di passare senza problemi tra esplorazione e sfruttamento.
- Con $\vec{A}$ decrescente, metà delle iterazioni sono dedicate all'esplorazione ($|\vec{A}| > 1$) e l'altra metà è dedicata allo sfruttamento ($|\vec{A}| < 1$).

- Il GWO ha solo due parametri principali da regolare (a e $\vec{C}$).

2.4 Fruit Fly Optimization Algorithm

Come illustrato in [11], il FOA è un algoritmo meta-euristico population-based, che ricade nella categoria degli algoritmi SI. Si basa sul comportamento di ricerca del cibo del moscerino della frutta.

Il moscerino della frutta è superiore ad altre specie nella percezione, specialmente nell'olfatto e nella visione. Gli organi dell'olfatto dei moscerini della frutta sono in grado di percepire tutti i tipi di odori nell'aria, può anche sentire l'odore di una fonte di cibo da 40 km di distanza. Quindi, dopo essersi avvicinato alla posizione della fonte di cibo, usando la sua visione sensibile è in grado di raggiungere l'ubicazione e volare in quella direzione.

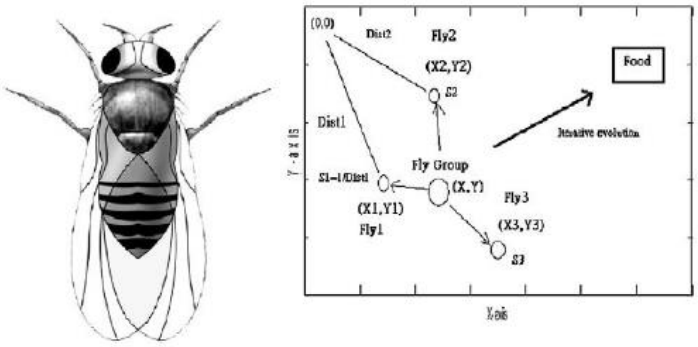


Fig. 8: Aspetto del corpo del moscerino della frutta e ricerca iterativa di cibo di un gruppo di moscerino della frutta.

Gli step del FOA sono i seguenti.

1. Viene inizializzata la posizione casuale dello sciame di moscerini della frutta
2. Si definisce la direzione e la distanza casuali per la ricerca delle fonti cibo, mediante l'olfatto di un singolo moscerino della frutta.

$$X_i = X_{axis} + RandomValue$$
$$Y_i = Y_{axis} + RandomValue$$

3. Poiché la posizione dell'alimento non può essere conosciuta, viene quindi stimata prima la distanza dall'origine ($Dist$), quindi viene calcolato il valore di valutazione della concentrazione dell'odore (S), e questo valore è il reciproco della distanza.

$$Dist_i = \sqrt{X_i^2 + Y_i^2}$$
$$S_i = {}^{1}/_{Dist_i}$$

4. Si sostituisce il valore di valutazione della concentrazione dell'odore (S) nella funzione di valutazione della concentrazione dell'odore (chiamata funzione di fitness) in modo da trovare la concentrazione dell'odore ($Smell$) della posizione individuale del moscerino della frutta.

$$Smell_i = Function(S_i)$$

5. Si identifica (con il suo indice) il moscerino della frutta con la massima concentrazione di odore (trovando il valore massimo) tra lo sciame di moscerini della frutta.

$$[BestSmell, BestIndex] = \max(Smell)$$

6. Si mantiene il miglior valore di concentrazione dell'odore e le coordinate x, y, e in questo momento lo sciame di moscerini della frutta utilizzerà la visione per volare verso quella posizione.

$$SmellBest = bestSmell$$
$$X_{axis} = X(bestIndex)$$
$$Y_{axis} = Y(bestIndex)$$

7. Si ripetono i passaggi 2-5. Se la concentrazione dell'odore è superiore alla precedente concentrazione dell'odore iterativo, ripetere il passaggio 6.

2.5 Colliding Bodies Optimization

Il CBO è un algoritmo meta-euristico population-based, che ricade nella categoria degli algoritmi che si basano sulle leggi della fisica. Infatti, si basa su collisioni

unidimensionali tra corpi, dove ogni oggetto o un corpo (soluzione agente) è dotato di massa e velocità, come riportato in [12].

Questa collisione fa sì che gli agenti si spostino verso posizioni migliori nello spazio di ricerca. Il CBO utilizza una formulazione semplice per trovare il minimo o il massimo delle funzioni e non dipende da alcun parametro interno.

2.5.1 La collisione tra corpi

Le collisioni tra corpi sono governate dalle leggi della quantità di moto e dell'energia. Quando si verifica una collisione in un sistema isolato, si conserva la quantità di moto totale del sistema di oggetti. A condizione che non vi siano forze esterne nette che agiscono sugli oggetti, la quantità di moto di tutti gli oggetti prima dell'urto è uguale alla quantità di moto di tutti gli oggetti dopo l'urto.

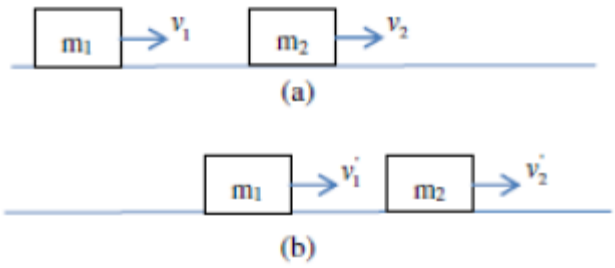


Fig. 9: La collisione tra due corpi. (a) prima dell'urto e (b) dopo l'urto.

La conservazione della quantità di moto totale richiede che la quantità di moto totale prima dell'urto sia uguale alla quantità di moto totale dopo l'urto e può essere espressa dalla seguente equazione:

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2'$$

Allo stesso modo, la conservazione dell'energia cinetica totale è espressa come:

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 v_1'^2 + \frac{1}{2} m_2 v_2'^2 + Q$$

dove v_1 è la velocità iniziale del primo oggetto prima dell'impatto, v_2 è la velocità iniziale del secondo oggetto prima dell'impatto, v_1' è la velocità finale del primo oggetto

dopo l'impatto, v_2' è la velocità finale del secondo oggetto dopo l'impatto, m_1 è la massa del primo oggetto, m_2 è la massa del secondo oggetto e Q è la perdita di energia cinetica dovuta all'impatto.

Le formule per le velocità dopo una collisione unidimensionale sono:

$$v_1' = \frac{(m_1 - \varepsilon m_2)v_1 + (m_2 + \varepsilon m_2)v_2}{m_1 + m_2}$$
$$v_2' = \frac{(m_2 - \varepsilon m_1)v_2 + (m_1 + \varepsilon m_1)v_1}{m_1 + m_2}$$

dove ε è il coefficiente di restituzione (COR) dei due corpi in collisione, definito come il rapporto tra la velocità relativa di separazione e la velocità relativa di avvicinamento:

$$\varepsilon = \frac{|v_2' - v_1'|}{|v_2 - v_1|} = \frac{v'}{v}$$

Secondo il coefficiente di restituzione, ci sono due casi speciali di qualsiasi collisione come segue:

1. Si definisce urto perfettamente elastico quello in cui non vi è perdita di energia cinetica nell'urto ($Q = 0$ ed $\varepsilon = 1$). In questo caso, dopo l'urto, la velocità di separazione è elevata. In realtà, qualsiasi collisione macroscopica tra oggetti convertirà una parte dell'energia cinetica in energia interna e altre forme di energia.
2. Un urto anelastico è quello in cui parte dell'energia cinetica viene cambiata in qualche altra forma di energia nell'urto. La quantità di moto si conserva negli urti anelastici (come per gli urti elastici), ma non è possibile tracciare l'energia cinetica attraverso l'urto poiché parte di essa verrà convertita in altre forme di energia. In questo caso, il coefficiente di restituzione non è uguale a uno ($Q \neq 0$ ed $\varepsilon \leq 1$) e la velocità di separazione è bassa.

Per gli oggetti reali, il valore di ε è compreso tra 0 e 1.

2.5.2 L'algoritmo CBO

In CBO, ogni soluzione candidata X_i, contenente un numero di variabili (cioè $x_i = \{x_{ij}\}$), è considerato come un corpo (chiamato d'ora in poi CB).

Gli oggetti sono composti da due gruppi principali uguali: corpi stazionari e corpi in movimento, dove gli oggetti in movimento si muovono per seguire oggetti fermi verificando così una collisione tra oggetti.

Questo viene fatto per due scopi:

1. migliorare le posizioni degli oggetti in movimento
2. spingere oggetti fermi verso posizioni migliori.

Dopo la collisione, le nuove posizioni dei corpi in collisione vengono aggiornate in base alla nuova velocità utilizzando le leggi di collisione.

Le fasi di tale meta-euristica sono le seguenti:

1. Le posizioni iniziali dei CB sono determinate con l'inizializzazione casuale di una popolazione di individui nello spazio di ricerca:

$$x_i^0 = x_{min} + rand(x_{max} - x_{min}), \quad i = 1, 2, \dots, n$$

dove x_i^0 determina il vettore del valore iniziale dell'i-esimo CB, x_{min} e x_{max} sono i vettori dei valori minimo e massimo ammissibili delle variabili, $rand$ è un numero casuale nell'intervallo $[0, 1]$ e n è il numero di CB.

2. La grandezza della massa corporea per ogni CB è definita come:

$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^{n} \frac{1}{fit(i)}}$$

dove $fit(i)$ rappresenta il valore della funzione obiettivo dell'agente i, mentre n è la dimensione della popolazione. Un CB con buoni valori ha una massa maggiore di quelli con valori meno buoni. Inoltre, per la massimizzazione, la funzione obiettivo $fit(i)$ sarà sostituita da $\frac{1}{fit(i)}$.

3. La disposizione dei valori della funzione obiettivo dei CB viene eseguita in ordine crescente. I CB ordinati sono equamente divisi in due gruppi.
   - La metà inferiore degli CB (CB stazionari): questi CB sono buoni agenti che sono stazionari e la velocità di questi corpi prima della collisione è zero, ovvero:

$$v_i = 0, \quad i = 1, \dots, \frac{n}{2}$$

   - La metà superiore dei CB (CB in movimento): questi CB si spostano verso la metà inferiore. Quindi, secondo la Fig. (b), i CB migliori e peggiori, cioè gli agenti con un valore di fitness superiore, di ciascun gruppo si scontreranno insieme.
     Il cambiamento della posizione del corpo rappresenta la velocità di questi corpi prima dell'urto come:

$$v_i = x_i - x_{i-\frac{n}{2}}, \quad i = \frac{n}{2} + 1, \dots, n$$

     Dove, v_i e x_i sono rispettivamente la velocità e il vettore posizione dell'i-esimo CB in questo gruppo e $x_{1-n/2}$ è l'i-esima posizione della coppia CB di x_i nel gruppo precedente.

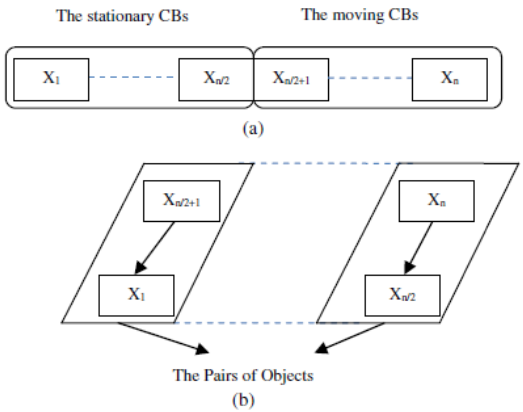


Fig.10: (a) CB ordinati in ordine crescente e (b) coppie di oggetti in collisione.

4. Dopo l'urto, le velocità dei corpi in collisione in ciascun gruppo vengono valutate utilizzando le equazioni di seguito, e la velocità prima dell'urto. La velocità di ciascun CB in movimento dopo l'urto è ottenuta da:

$$v_i' = \frac{(m_i - \varepsilon m_{i-\frac{n}{2}})v_i}{m_i + m_{i-\frac{n}{2}}}, \quad i = \frac{n}{2} + 1, \dots, n$$

dove v_i e v_i' sono rispettivamente la velocità dell'i-esimo CB in movimento prima e dopo l'urto, m_i è la massa dell'i-esimo CB, $m_{i-n/2}$ è la massa della i-esima coppia di CB. Inoltre, la velocità di ciascun CB stazionario dopo l'urto è:

$$v_i' = \frac{(m_{i+\frac{n}{2}} + \varepsilon m_{i+\frac{n}{2}})v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}}, \quad i = 1, \dots, \frac{n}{2}$$

dove $v_{i+\frac{n}{2}}$ e v_i' sono rispettivamente la velocità dell'i-esima coppia di CB in movimento prima e l'i-esimo CB stazionario dopo l'urto, m_i è la massa dell'i-esimo CB, $m_{i+\frac{n}{2}}$ è la massa dell'i-esima coppia di CB in movimento ed ε è il valore del parametro COR la cui legge di variazione sarà discussa nella prossima sezione.

5. Le nuove posizioni di CB vengono valutate utilizzando le velocità generate dopo la collisione in posizione di CB stazionari. Le nuove posizioni di ogni CB in movimento sono:

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \otimes v_i', \quad i = \frac{n}{2} + 1, \dots, n$$

dove x_i^{new} e v'_i sono rispettivamente la nuova posizione e la velocità dopo l'urto dell'i-esimo CB in movimento, mentre $x_{i-n/2}$ è la vecchia posizione dell'i-esima coppia di CB stazionari. Inoltre, le nuove posizioni degli CB stazionari sono ottenute da:

$$x_i^{new} = x_i + rand \otimes v_i', \quad i = 1, \dots, \frac{n}{2}$$

dove x_i^{new}, x_i e v'_i sono rispettivamente la nuova posizione, la vecchia posizione e la velocità dopo l'urto dell'i-esimo CB stazionario, $rand$ è un vettore casuale uniformemente distribuito nell'intervallo (0,1) e il simbolo $\otimes$ denota una moltiplicazione elemento per elemento.

6. L'ottimizzazione viene ripetuta dal passaggio 2, fino a quando non viene soddisfatto un criterio di terminazione, come il numero massimo di iterazioni. Va notato che lo stato di un corpo (fermo o in movimento) e la sua numerazione vengono modificati in due successive iterazioni.

Possiamo illustrare anche il diagramma che descrive il funzionamento:

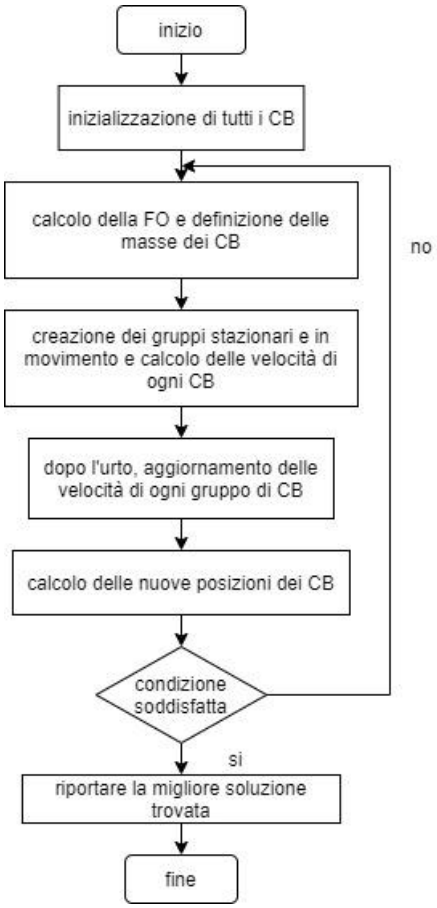


Fig. 11: diagramma di flusso che descrive il funzionamento del CBO

2.5.3 Il coefficiente di restituzione (COR)

Nelle meta-euristiche, il processo di ricerca è suddivisibile in due fasi: l'esplorazione dello spazio di ricerca e sfruttamento delle migliori soluzioni trovate, ed è molto importante avere un adeguato equilibrio tra esplorazione e sfruttamento. Infatti, nel processo di ottimizzazione, l'esplorazione dovrebbe essere ridotta gradualmente mentre contemporaneamente dovrebbe essere aumentato lo sfruttamento.

A questo scopo, nel CBO viene introdotto un indice in termini di coefficiente di restituzione (COR) per controllare il tasso di esplorazione e sfruttamento. Questo indice è definito come il rapporto tra la velocità di separazione di due agenti dopo l'urto e la velocità di avvicinamento di due agenti prima dell'urto.

2.6 Algoritmi utilizzati

Nel resto della tesi sono state adottate le seguenti implementazioni degli algoritmi euristici menzionati:

- Artificial Bee Colony [13]
- Particle swarm optimization [14]
- Fruit Fly Optimization Algorith [15]
- Colliding Bodies Optimization [16]
- Grey Wolf Optimizer [17]

Capitolo 3

Caso di studio: L'Esarotore

In questo capitolo, si introduce il caso di studio per il quale si intendono sfruttare gli algoritmi di SI finora presentati. Il problema da risolvere è la taratura dei parametri di controllo di un esarotore con configurazione inclinata dei motori. In primo luogo, si introduce il modello dinamico del drone. Successivamente, si mostra lo schema di controllo sul quale verrà eseguito in simulazione il processo di taratura del controllore. Infine, si definiscono gli indici di prestazione che verranno usati come funzioni di fitness nei vari algoritmi di SI.

3.1 Il Modello

Un esarotore inclinato è un caso speciale dei Generically Tilted Multirotor (GTM) recentemente introdotti [18]. Di seguito vengono brevemente presentate la cinematica e la dinamica dei GTM; quindi, vengono dettagliati gli effetti dell'inclinazione del motore; infine, viene discussa la modellazione dei guasti dell'attuatore.

3.1.1 Cinematica e dinamica

Si considerano due sistemi di riferimento: uno solidale alla Terra $R_E - \{O_E;\ x_E;\ y_{E;}\, z_E\}$ che si presume sia inerziale, ed uno solidale al drone $R_B - \{O_B;\ x_B;\ y_{B;}\, z_B\}$, dove O_B coincide con il centro di massa del drone e gli assi sono orientati rispettivamente in avanti, a sinistra ed in alto. Supponendo che non vi siano disturbi esterni e che la struttura meccanica sia simmetrica, un modello GTM può essere rappresentato come

$$m\ddot{p}_F = -mge_3 + Rv_1$$

$$J\dot{\omega} = -\omega \times J\omega + v_2$$
$$\dot{R} = RS(\omega)$$

dove $p_F = [x_F, y_F, z_F]^T$ è la posizione del centro di massa in R_E, $\omega = [p\ q\ r]^T$ è un vettore composto dalle velocità di rotazione istantanee attorno agli assi di R_B, $R \in SO(3)$ è la matrice di rotazione che mappa un vettore espresso in riferimento a R_B nello stesso vettore espresso in riferimento a R_E, m è la massa totale del sistema, g è l'accelerazione gravitazionale, $e_3 = [0\ 0\ 1]^T$ indica la direzione verso l'alto in R_E e $J = diag(I_x, I_y, I_z)$ è la matrice di inerzia, dove I_x, I_y e I_z sono i momenti di inerzia lungo gli assi x_B, y_B e z_B.

La funzione $S: \mathbb{R}^3 \to SO(3)$ è data da

$$S(\omega) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

Il vettore $v_1 = [v_x, v_y, v_z]^T$ è un vettore tridimensionale che rappresenta le forze agenti sul telaio dovute ai motori, mentre $v_2 = [v_p, v_q, v_r]^T$ è un vettore tridimensionale composto dalle coppie.

Adottando la notazione roll-pitch-yaw, la matrice di rotazione R è parametrizzata come

$$R = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\varphi s_\theta - c_\varphi s_\psi & s_\varphi s_\psi + c_\varphi c_\psi s_\theta \\ c_\theta s_\psi & c_\varphi c_\psi + s_\varphi s_\psi s_\theta & c_\varphi s_\psi s_\theta - c_\psi s_\varphi \\ -s_\theta & c_\theta s_\varphi & c_\varphi c_\theta \end{bmatrix}$$

dove φ, θ e ψ sono gli angoli di roll, pitch, e yaw (rollio, beccheggio e imbardata) e vengono adottate le notazioni brevi $c(\cdot) = \cos(\cdot)$ e $s(\cdot) = \sin(\cdot)$.
Indicando con r_{ij} le componenti della matrice di rotazione R, cioè $R = [r_{ij}]$ con $i, j = 1, 2, 3$, il modello può essere rappresentato localmente come:

$$m\ddot{x}_F = r_{11}v_x + r_{12}v_y + r_{13}v_z$$
$$m\ddot{y}_F = r_{21}v_x + r_{22}v_y + r_{23}v_z$$
$$m\ddot{z}_F = r_{31}v_x + r_{32}v_y + r_{33}v_z - mg$$
$$I_x\dot{p} = -qr(I_z - I_y) + v_p$$

$$\dot{q} = -pr(I_x - I_z) + v_q$$
$$I_z\dot{r} = -pq(I_y - I_x) + v_r$$
$$\dot{\varphi} = p + q\sin(\varphi)\tan(\theta) + r\cos(\varphi)\tan(\theta)$$
$$\dot{\theta} = q\cos(\varphi) - r\sin(\varphi)$$
$$\dot{\psi} = [q\sin(\varphi) + r\cos(\varphi)]/\cos(\theta)$$

Da notare che i droni più comuni hanno pale molto leggere e a bassa inerzia, in quanto tale l'effetto giroscopico è trascurato nel modello proposto, come esposto in [19].

3.1.2 Inclinazione del rotore

Il modo in cui v_1 e v_2 sono correlati agli ingressi di controllo effettivi dipende dalla configurazione del rotore. Per descrivere il modello di un esarotore GTM parametrizzato si utilizzano tre angoli: α e β per esprimere l'inclinazione dei rotori, cioè rispettivamente una rotazione laterale e una rotazione interna, mentre l'angolo γ atto a definire la disposizione delle eliche.

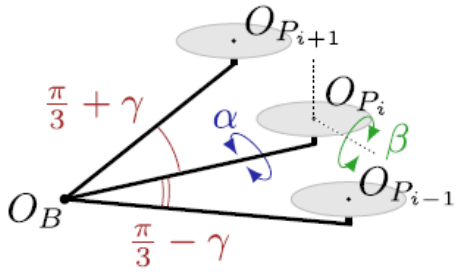


Fig. 12: Disposizione di tre eliche consecutive, evidenziando l'effetto degli angoli α, β e γ.

Si assume che l'orientamento (ovvero α e β) di ciascun motore, così come la sua posizione in R_B, sia fisso e noto. Inoltre, si presume che gli stessi angoli α e β descrivano l'orientamento di tutti i rotori, cioè tutti i rotori hanno lo stesso α e β e lo stesso orientamento.

I sei attuatori sono complanari ed equidistanti su un cerchio di raggio l attorno al centro di massa e numerati in senso orario, da 1 a 6, come mostrato nella seguente figura.

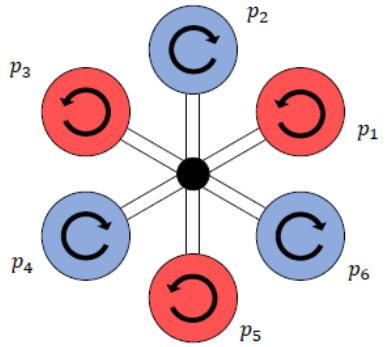


Fig. 13: Configurazione dei rotori dell'esarotore

A seconda di α e β, ogni elica produce una forza $f_i \in \mathbb{R}^3$ che ha componenti non nulle lungo tutti e tre gli assi x_B, y_B e z_B. Questo rende i GTM più generici dei droni collineari, dove la forza viene esercitata solo nella direzione di z_B.

Si indica con $u_i \in \mathbb{R}$ il modulo della forza f_i, con $i = 1, \dots, 6$ nel caso dell'esarotore, dove u_i è assunto positivo/negativo se f_i ha una componente z_B positiva/negativa, che è ancora la componente primaria, per la necessità di contrastare la forza di gravità.

Si definisce, inoltre, anche il vettore $u_i = [u_1, \dots., u_6]^{\mathrm{T}}$ che rappresenta le effettive grandezze delle forze. La relazione tra v_1, v_2 e u è della forma:

$$v_1 = F_1 u \qquad\qquad v_2 = F_2 u$$

Dove F_1 e $F_2 \in \mathbb{R}^{3x6}$ sono, rispettivamente, le matrici di input della forza e delle coppie di controllo. È da notare che sia F_1 che F_2 dipendono dalla scelta di α e β, nonché dai coefficienti di spinta e resistenza delle eliche e dalla lunghezza del braccio.

Le equazioni (5) possono essere riscritte, in modo compatto, come:

$$v \triangleq \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} u = Fu$$

Dove il vettore a sei dimensioni v è considerato come vettore di input virtuale.

In tutto questo elaborato, α e β sono sempre considerati tali che il GTM è pienamente attuato, il che vuol dire rango $F = 6$. Mentre, per una configurazione standard (coplanare e

collineare) con più di quattro attuatori, gli angoli di inclinazione sono posti come $\alpha = \beta = 0$ e rango $F = 4$, come detto in [19].

3.2 Leggi di controllo

A questo punto è necessario progettare le legge di controllo del controllore Inner Loop e dell'Outer Loop che sono state utilizzate per realizzare le simulazioni.

Come definito in [19], per prima cosa, si introduce il vettore $\eta = [\varphi\ \theta\ \psi]^T$ che permette di riscrivere il modello (1) precedentemente definito come:

$$m\ddot{p}_F = -mge_3 + Rv_1$$
$$J\dot{\omega} = -\omega\, x\, J\omega + v_2$$
$$\eta = \dot{T}(\eta)\omega$$

Dove

$$T(\eta) = \begin{bmatrix} 1 & s_\varphi t_\theta & c_\varphi t_\theta \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi/c_\theta & c_\varphi/c_\theta \end{bmatrix}$$

3.2.1 Progettazione della legge di controllo Inner Loop

Si esegue una feedback linearization per riscrivere il modello in modo più conveniente e si pone $\xi_1 = [z_F^T, \eta^T]$ e $\xi_2 = [\dot{z}_F^T, (T(\eta)\omega)^T]$. Quindi, la dinamica dell'altitudine e dell'assetto dell'esarotore inclinato può essere riscritta come

$$\dot{\xi}_1 = \xi_2$$

$$\dot{\xi}_2 = b(\xi_1, \xi_2) + A(\xi_1)FWu$$

Dove:

$$b(\xi_1, \xi_2) = \begin{bmatrix} -mg \\ \dot{T}(\eta)\omega - T(\eta)J^{-1}(\omega \times J\omega) \end{bmatrix}$$

$$A(\xi_1) = \begin{bmatrix} \frac{1}{m}e_3^T R(\eta) & 0 \\ 0 & T(\eta)J^{-1} \end{bmatrix}$$

La legge di controllo è impostata come:

$$u = \widehat{W}^{-1}(A(\xi_1)F)^{+}v$$

Dove v è selezionata come:

$$v = -b(\xi_1, \xi_2) + \ddot{\xi}_r - \alpha_1(\xi_1 - \dot{\xi}_r) - \alpha_0(\xi_1 - \xi_r)$$

Si noti che le matrici α_0 e α_1 sono diagonali, ovvero $\alpha_0 = diag(\alpha_{0,1}, \ldots., \alpha_{0,4})$ e $\alpha_1 = diag(\alpha_{1,1}, .., \alpha_{1,4})$, e queste sono le matrici costituite dai parametri da tarare in modo tale da controllare la traiettoria dell'esarotore, come scritto in [19].

3.2.2 Progettazione della legge di controllo Outer loop

Per realizzare questa legge di controllo, in questo caso, ci si basa sull'assunzione di near hover, ovvero si suppone che $\varphi \approx 0$ e $\theta \approx 0$.

Sia r_{ij} l'elemento di $R(\eta)$ che appartiene alla i-esima riga e alla j-esima colonna (ovvero $r_{ij} = e_{ij}^T R(\eta) e_j$. Inoltre, siano v_x, v_y e v_z le componenti delle forze complessive prodotte dai motori. La dinamica della posizione lineare orizzontale può essere approssimata come

$$\begin{bmatrix} \ddot{x}_F \\ \ddot{y}_F \end{bmatrix} = \frac{1}{m}\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}\begin{bmatrix} v_x \\ v_y \end{bmatrix} + \frac{v_z}{m}\begin{bmatrix} s_\varphi s_\psi + c_\varphi c_\psi s_\theta \\ c_\varphi s_\psi s_\theta - c_\psi s_\varphi \end{bmatrix}$$

$$\approx \frac{1}{m}\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}\begin{bmatrix} v_x \\ v_y \end{bmatrix} + \frac{v_z}{m}\begin{bmatrix} \sin(\psi) & \cos(\psi) \\ -\cos(\psi) & \sin(\psi) \end{bmatrix}\begin{bmatrix} \varphi \\ \theta \end{bmatrix}$$

I riferimenti φ_r e θ_r per l'inner loop, per convergere asintoticamente al riferimento di posizione x_{Fr} e y_{Fr} sono dati da

$$\begin{bmatrix} \varphi_r \\ \theta_r \end{bmatrix} = \frac{m}{v_z}\begin{bmatrix} \sin(\psi) & -\cos(\psi) \\ \cos(\psi) & \sin(\psi) \end{bmatrix}\left(-\frac{1}{m}\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}\begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}\right)$$

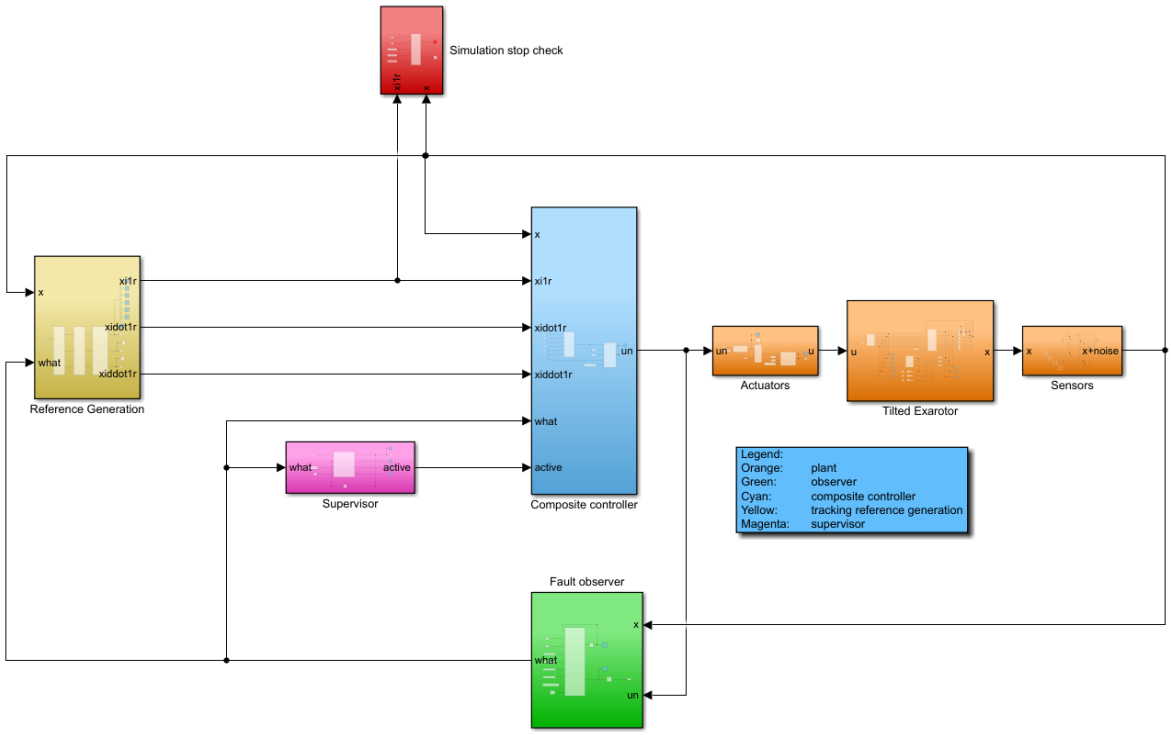
In particolare:

$$\tau_x = \ddot{x}_{Fr} - \alpha_{1x}(\dot{x}_F - \dot{x}_{Fr}) - \alpha_{0x}(x_F - x_{Fr})$$
$$\tau_y = \ddot{y}_{Fr} - \alpha_{1y}(\dot{y}_F - \dot{y}_{Fr}) - \alpha_{0y}(y_F - y_{Fr})$$

Dove α_{1x}, α_{0x}, α_{1y} e α_{0y} sono parametri che vengono scelti in modo opportuno per avere una dinamica di posizione stabile. Tali parametri sono oggetto di taratura dagli algoritmi euristici, come scritto in [19].

3.3 Schema di controllo

Di seguito viene riportato lo schema di controllo realizzato in Simulink dell'esarotore con cui sono state realizzate le simulazioni



Dove nel dettaglio, i blocchi arancioni rappresentano il drone nelle sue componenti:

- Il blocco "Actuators" modella i limiti fisici degli attuatori (saturazione) e gli eventuali guasti dei motori;
- Il blocco "Tilted Exarotor" contiene la configurazione degli attuatori, che lega la forza generata dai singoli motori alle forze e coppie virtuali, ed il modello cinematico e dinamico del drone, che descrive il moto del corpo soggetto a tali forze e coppie.
- Il blocco "Sensors" modella la rumorosità di una IMU comunemente impiegata in droni commerciali (TDK InvenSense Mpu-9250).

Mentre:

- il blocco in verde “Fault Observer” rappresenta un osservatore degli eventuali guasti dell’ingresso,
- la stima del guasto è fornita al blocco in magenta “Supervisor”, che gestisce lo spegnimento degli eventuali motori inservibili.
- il blocco in giallo “Reference Generation” contiene diversi metodi per la generazione del riferimento di volo.
- il blocco in rosso “Simulation Stop Check” termina in anticipo la simulazione qualora il volo sia fallito, cioè quando l’errore di tracking supera una soglia massima.
- il blocco in celeste “Composite Controller” contiene il controllore di cui si desidera fare il tuning.

Si vuole sottolineare che, ai fini della taratura del controllore, si realizzano le seguenti modifiche allo schema precedentemente illustrato:

1. Il rumore aggiunto dal sensore ha un seed fissato, per evitare variazioni randomiche e non ripetibili.
2. I guasti dei motori non sono considerati, quindi i blocchi in verde e viola non hanno alcun effetto.

3.4 Traiettoria

In questo sotto-capitolo verrà illustrata la traiettoria scelta da far realizzare al drone in esame. Per poterla attuare, il riferimento utilizzato per la taratura è descritto da:

```
xFr = Ax*sin((2*pi*t)/T);
yFr = Ay*sin((4*pi*t)/T);
zFr = kz*t;
psir = kpsi*t;
```

Il meccanismo è quindi il seguente, una volta fissata un’ampiezza orizzontale (A_x), un’ampiezza verticale (A_y), un periodo (T), un rate di salita (k) ed un rate di rotazione ($kpsi$). L’obiettivo è quello di ottenere un riferimento a forma di “otto” o “infinito” nelle variabili x ed y, mentre la quota e l’angolo di yaw crescono linearmente nel tempo. Si vuole sottolineare che tale riferimento è fissato per la taratura di tutte le simulazioni, in modo da poter realizzare un confronto esaustivo.

3.5 Indici

Gli indici testati per la taratura degli algoritmi meta-euristici sono quattro: IAE, ISE, ISE con in più la variazione dello sforzo di controllo ed infine ISE considerando anche l'inner loop.

Tali indici sono calcolati come segue:

$$e_x = x(:,2) - xF_r(:,2);$$
$$e_y = x(:,4) - yF_r(:,2);$$
$$e_z = x(:,6) - zF_r(:,2);$$
$$e_{phi} = x(:,11) - phi_r(:,2);$$
$$e_{theta} = x(:,12) - theta_r(:,2);$$
$$e_{psi} = x(:,13) - psi_r(:,2);$$

$$IAE_x = sum(|e_x|) * T_c;$$
$$IAE_y = sum(|e_y|) * T_c;$$
$$IAE_z = sum(|e_z|) * T_c;$$
$$IAE_{psi} = sum(|e_{psi}|) * T_c;$$

$$ISE_x = sum(e_x^2) * T_c;$$
$$ISE_y = sum(e_y^2) * T_c;$$
$$ISE_z = sum(e_z^2) * T_c;$$
$$ISE_{phi} = sum(e_{phi}^2) * T_c;$$
$$ISE_{theta} = sum(e_{theta}^2) * T_c;$$
$$ISE_{psi} = sum(e_{psi}^2) * T_c;$$

$$deltaU = u(2:end,:) - u(1:end-1,:);$$
$$U = sum\left(sum(deltaU(:,2:end))^2\right) * T_c;$$

$$IAE = IAE_x + IAE_y + IAE_z + IAE_{psi};$$

$$ISE = ISE_x + ISE_y + ISE_z + ISE_{psi};$$

$$ISE_{deltaU} = \frac{ISE_x + ISE_y + ISE_z + ISE_{psi} + U}{100};$$

$$ISE_{innerouter} = ISE_x + ISE_y + ISE_z + peso_{inner} * ISE_{phi} + peso_{inner} * ISE_{theta} + ISE_{psi};$$

Per prima cosa, si calcolano gli errori di tracking, relativi alle 6 variabili decisionali, ovvero $e_x, e_y, e_z, e_\varphi, e_\theta$ ed e_ψ, che vengono calcolati come differenza in ogni istante tra lo stato effettivo del drone ed il rispettivo stato di riferimento desiderato.

Successivamente, si determina l'integrale dell'errore assoluto, ovvero IAE_x, IAE_y, IAE_z e IAE_ψ, di ciascuna variabile di interesse come sommatoria degli errori di tracking in valore assoluto. Nello stesso modo, si calcola l'integrale dell'errore quadratico, cioè $ISE_x, ISE_y, ISE_z, ISE_\varphi, ISE_\theta$ ed ISE_ψ, delle variabili in esame come sommatoria degli errori di tracking elevati al quadrato.

Inoltre, si definiscono le variazioni dello sforzo di controllo $deltaU$ come differenza tra gli sforzi di controllo da un istante all'altro, che verrà utilizzato per poi calcolare un indice quadratico U di tale quantità in maniera simile a quanto fatto all'ISE. Per concludere, le variabili determinate finora vengono utilizzate per ottenere i quattro indici necessari per realizzare la taratura dell'algoritmo tramite semplici somme pesate.

Capitolo 4

Risultati e Confronto tra algoritmi

Durante lo studio preliminare delle meta-euristiche descritte nel capitolo precedente, queste sono state testate con funzioni di benchmark, in modo tale da valutare e confrontare le prestazioni dei paradigmi. Le funzioni di test utilizzate sono la funzione DeJong di quinto grado e la funzione Sphere, le quali verranno descritte brevemente di seguito. Nella prima parte di questo capitolo si riporta un confronto sulle prestazioni degli algoritmi su queste due funzioni e verranno esposti e commentati i risultati ottenuti.

Una volta realizzato uno studio preliminare per comprendere il comportamento degli algoritmi di ottimizzazione in esame, tali algoritmi sono stati applicati al caso di studio, ovvero il drone esarotore. Nella seconda parte di questo capitolo sono descritti e commentati i risultati finali delle varie simulazioni, con annesse delle osservazioni. In particolare, sono state sperimentate due tipologie di simulazione, una con un time-out pari 5 minuti (300 secondi) e un time-out pari a 1 ora (3600 secondi). Nell'ultima parte del capitolo, infine, si riportano alcune modifiche al paradigma del PSO e i relativi risultati.

4.1 Test Preliminari

4.1.1 Funzione Sphere

La funzione Sphere, o anche chiamata funzione 1 di DeJong, è una funzione di benchmark semplice. Essa è continua, convessa e unimodale [20]. Ha la seguente definizione generale:

$$f(x) = \sum_{i=1}^{n} x_i^2$$

La funzione viene solitamente valutata sull'iper-cubo definito da $-5.12 \leq x_i \leq 5.12$, dove $i = 1, \ldots, n$. Il minimo globale $f(x) = 0$ è ottenibile per $x_i = 0, i = 1, \ldots, n$.

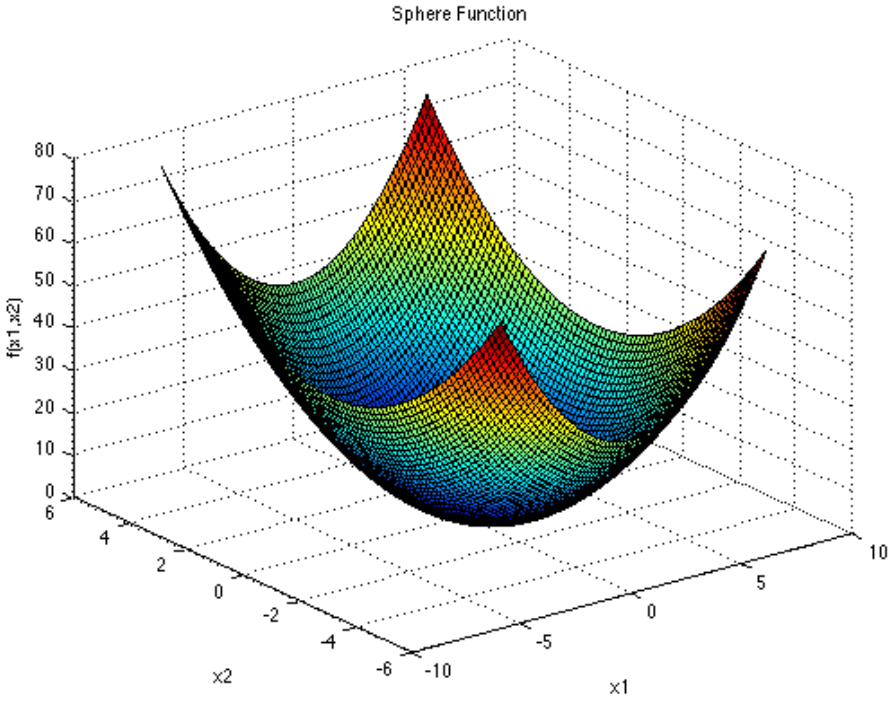


Fig. 14: Grafico 3D della funzione Sphere

4.1.2 Funzione di DeJong di quinto grado

La funzione di DeJong di quinto grado è una funzione di test multimodale che ha solo due variabili indipendenti, x_1 e x_2 [20]. Essa ha la seguente definizione:

$$f(x_1, x_2) = \left\{ 0.002 + \sum_{j=1}^{25} \left[j + \left(x_1 - a_{1j}\right)^6 + \left(x_2 - a_{2j}\right)^6 \right]^{-1} \right\}^{-1}$$

Dove:

$$\left(a_{ij}\right) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$$

La funzione può anche essere riscritta come segue:

$$f(x_1, x_2) = \left\{ 0.002 + \sum_{i=-2}^{2} \sum_{j=-2}^{2} [5(i+2) + j + 3 + (x_1 - 16j)^6 + (x_2 - 16i)^6]^{-1} \right\}^{-1}$$

È da sottolineare che la funzione viene solitamente valutata sul quadrato definito da $-65.536 \leq x_1 \leq 65.536\, e\, -65.536 \leq x_2 \leq 65.536$.

Come già detto, la funzione di DeJong è una funzione bidimensionale, con cadute molto brusche su una superficie prevalentemente piana e possiede ben 25 minimi locali. Il grafico di tale funzione è mostrato di seguito, da cui si può notare che non è chiaro quale di questi minimi locali sia il minimo globale.

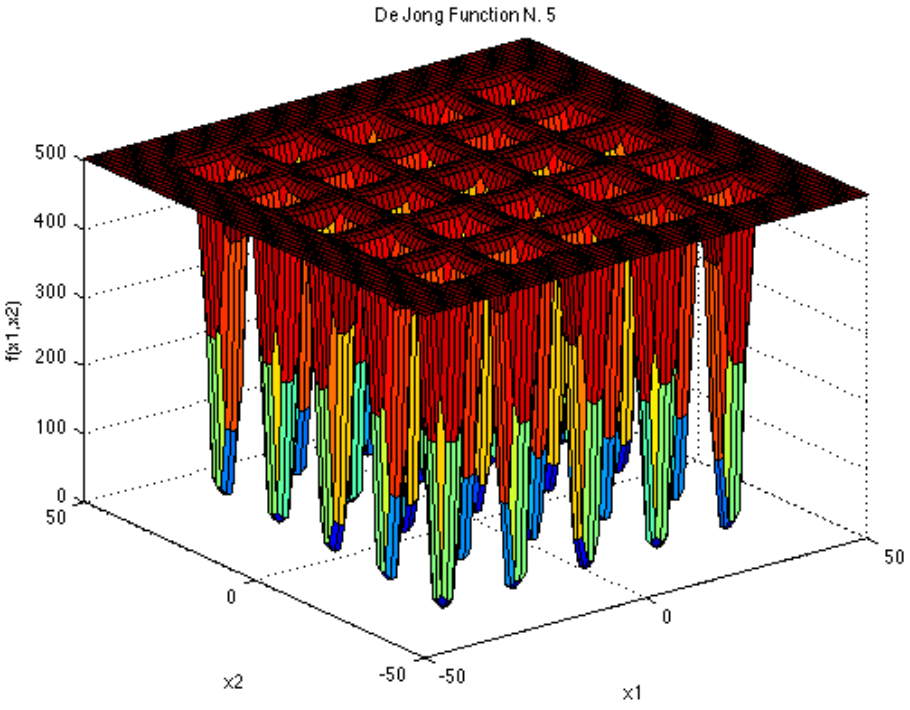


Fig. 15: Grafico 3D della funzione DeJong di quinto grado

4.1.3 Confronto su funzioni Sphere e DeJong

I cinque algoritmi meta-euristici sono stati inizialmente testati con la funzione di benchmark Sphere. Per avere un confronto esaustivo, le simulazioni vengono realizzate variando le condizioni di esecuzione. I test sono svolti a parità di condizioni, utilizzando una dimensione della popolazione uguale per tutti gli algoritmi. In particolare, PSO, GWO, FOA e CBO hanno la stessa dimensione della popolazione. La stessa popolazione è invece divisa equamente tra api operaie e spettatrici nell'algoritmo ABC. In questo modo, fissato il numero di iterazioni, tutti i test impiegano un tempo comparabile.

Test su funzione Sphere

1. Test con 50 iterazioni, lower bound posto a -10, upper bound posto a +10, 5 variabili decisionali e una popolazione pari a 100 elementi. I risultati sono descritti di seguito:

ABC	PSO	GWO	FOA	CBO
3.4212e-05	0.00072495	3.9729e-17	0.0063292	0.00027819

Tutti gli algoritmi convergono verso l'ottimo globale, ma con una diversa accuratezza. Si nota che il GWO ha le migliori performance, seguito dall'ABC. Il FOA presenta la soluzione peggiore.

2. Test con 100 iterazioni, lower bound posto a -100, upper bound posto a 100, con 5 variabili decisionali e una popolazione pari a 100 elementi. I risultati sono descritti di seguito:

ABC	PSO	GWO	FOA	CBO
6.3433e-10	5.8775e-09	7.8995e-28	5.3249e-05	5.7869e-05

Come nel caso precedente, il GWO ha le migliori prestazioni, seguito da ABC e PSO. FOA e CBO producono una soluzione peggiore. Si sottolinea che tutte le soluzioni sono migliorate con l'aumentare delle iterazioni.

3. Test con 200 iterazioni, lower bound posto a -100, upper bound posto a 100, con 5 variabili decisionali e una popolazione pari a 100 elementi. I risultati sono descritti di seguito:

ABC	PSO	GWO	FOA	CBO
7.1271e-20	4.639e-41	9.0134e-64	6.4643e-05	1.3808e-05

Anche in questo caso, le conclusioni sono consistenti: GWO, PSO e ABC restituiscono la soluzione ottima con un'incertezza inferiore alla precisione macchina. Al contrario, FOA e CBO non presentano miglioramenti rispetto al test con 100 iterazioni.

Test su funzione DeJong di quinto grado

Successivamente è stata testata la funzione di DeJong di quinto grado. Anche in questo caso, per poter realizzare un confronto esauriente, sono stati svolti tre test:

1. Test con 50 iterazioni, lower bound posto a -10, upper bound posto a +10, 2 variabili decisionali e una popolazione pari a 100 elementi:

ABC	PSO	GWO	FOA	CBO
0.9982	12.6705	12.6705	12.6705	12.6705

L'algoritmo ABC è l'unico ad avvicinarsi alla soluzione ottima, mentre gli altri algoritmi rimangono bloccati in minimi locali.

2. Test con 100 iterazioni, lower bound posto a -100, upper bound posto a 100, con 2 variabili decisionali e una popolazione pari a 100 elementi:

ABC	PSO	GWO	FOA	CBO
0.998	0.998	1.992	12.6705	0.998

Aumentando le iterazioni a 100 e aumentando anche i bound, ABC, PSO e CBO forniscono una buona approssimazione dell'ottimo globale, mentre GWO e FOA sono fermi su altri minimi locali.

3. Test con 200 iterazioni, lower bound posto a -200, upper bound posto a 200, con 2 variabili decisionali e una popolazione pari a 100 elementi:

ABC	PSO	GWO	FOA	CBO
0.998	0.998	0.998	12.6705	0.998

Aumentando le iterazioni a 200, tutti gli algoritmi forniscono una buona soluzione, tranne il FOA che rimane ancora bloccato in un ottimo locale.

4.2 Applicazione degli algoritmi al drone esarotore

Una volta realizzato uno studio preliminare per valutare il funzionamento degli algoritmi, questi sono stati testati sul modello effettivo del drone che è stato scelto come caso di studio.

A questo scopo sono state eseguite, a parità di condizioni, 15 simulazioni per ogni meta-euristica per poter realizzare una comparazione esaustiva, scegliendo di inserire un time-out di 300 secondi (5 minuti) in modo tale da avere un criterio per il raffronto.

Le condizioni iniziali utilizzate per l'esecuzione degli algoritmi, sono:

- Tempo massimo di esecuzione (time-out) pari a 300 secondi;
- Numero massimo di iterazioni pari a 100;
- Numerosità della popolazione degli agenti pari a 5, per l'Artificial Bee Colony (5 operaie e 5 spettatrici);
- Numerosità della popolazione degli agenti pari a 10, per i restanti algoritmi;
- Lower bound per le variabili decisionali pari a 0.0001;
- Upper bound per le variabili decisionali pari a 100.

Di seguito sono ripotate le migliori soluzioni trovate da ogni algoritmo, con IAE scelto come funzione per l'ottimizzazione:

ABC	PSO	GWO	FOA	CBO
23.65	24.83	25.72	20.09	21.99

Si vuole, inoltre, far notare che durante l'esecuzione delle varie simulazioni sono stati rilevati alcuni valori di IAE estremi, pari a 1˙000˙000˙000 (1e9), dovuti al verificarsi di un errore di tracking eccessivo, di conseguenza la simulazione veniva terminata in anticipo e veniva e riportato un valore di IAE alto.

Dai risultati ottenuti, utilizzando un time-out di 5 minuti, si può evincere che tra i cinque algoritmi di ottimizzazione, quello che fornisce il valore migliore è il Fruit Fly Optimization (FOA) ottenuto dopo 22 iterazioni a fronte di un numero massimo di 100 iterazioni, seguito dal CBO la cui soluzione ottima è ottenuta dopo 15 iterazioni, dall'ABC dopo 20 iterazioni, il PSO dopo appena 4 iterazioni ed infine il GWO dopo 17 iterazioni.

Al fine di avere due ulteriori parametri per il confronto delle meta-euristiche, al termine delle 15 simulazioni, sono state calcolate media e varianza delle soluzioni ottenute da ogni algoritmo. Per ridurre l'effetto dei valori estremi (1e9), questi non sono stati presi in considerazione nel calcolo, in modo tale che non vadano ad influenzare e contaminare i valori di media e varianza. I risultati ottenuti sono i seguenti:

	IAE
ABC	Media: 59.5599 Varianza: 4762.9544
PSO	Media: 39.7642 Varianza: 1445.8548
GWO	Media: 60.2868 Varianza: 2297.0669
FOA	Media: 46.2501 Varianza: 1920.8771
CBO	Media: 36.4971 Varianza: 473.3505

Nelle Fig. 16-20 sono riportati i grafici di evoluzione dell'IAE al variare delle iterazioni per ogni algoritmo impiegato. Come nel caso del calcolo della media e della varianza, per ridurre l'effetto dei valori estremi (1e9), questi non sono stati presi in considerazione nella generazione del grafico. In ciascun grafico, ogni linea, contrassegnata da un colore diverso, rappresenta una singola simulazione delle 15 totali, ottenuta dall'esecuzione di ogni paradigma.

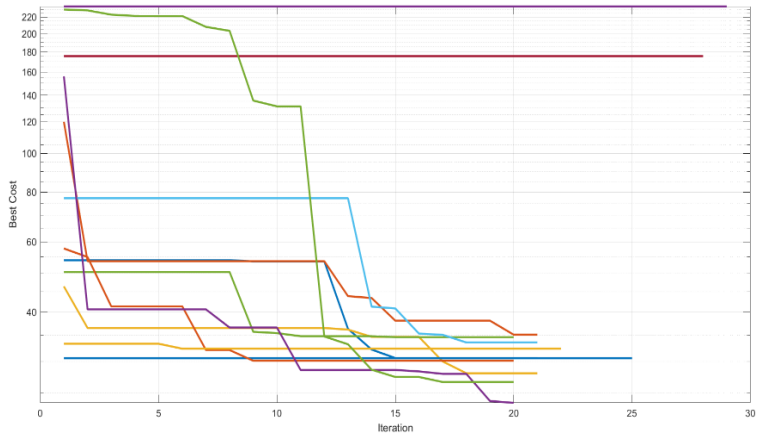


Fig. 16: Grafico dell'evoluzione IAE ottenuto con l'algoritmo ABC

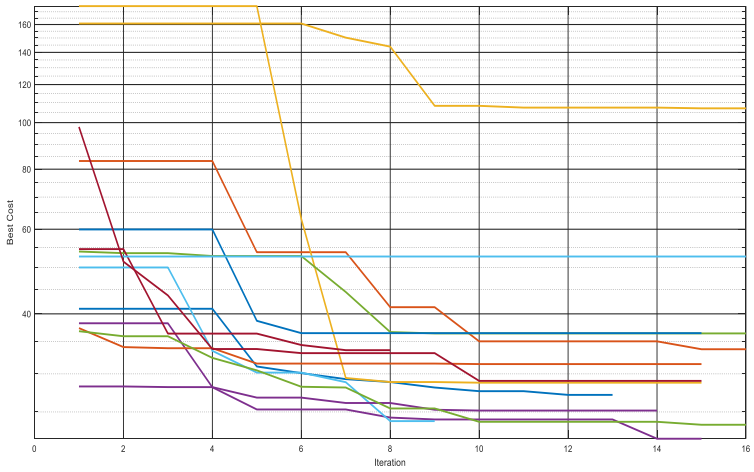


Fig. 17: Grafico dell'evoluzione IAE ottenuto con l'algoritmo CBO

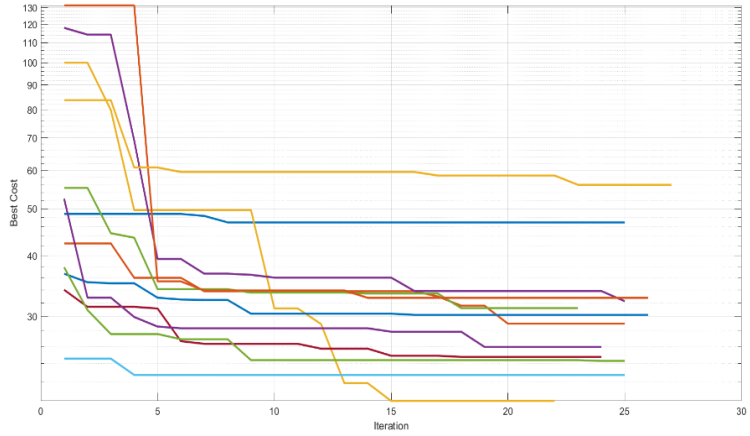


Fig. 18**:** Grafico dell'evoluzione IAE ottenuto con l'algoritmo FOA

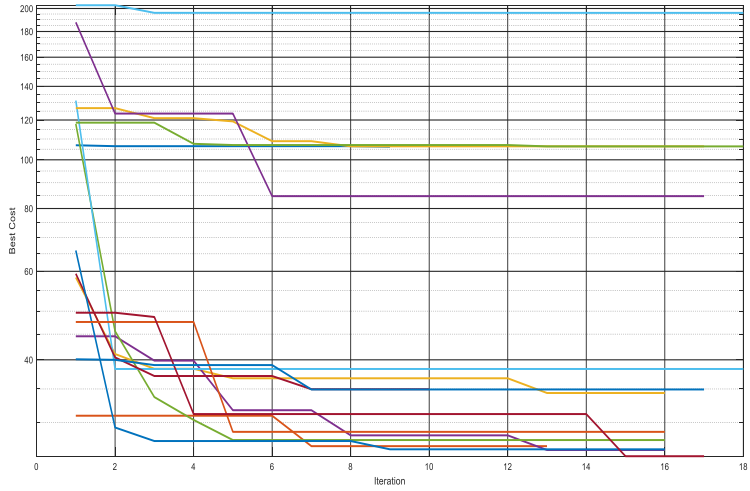


Fig. 19: Grafico dell'evoluzione IAE ottenuto con l'algoritmo GWO

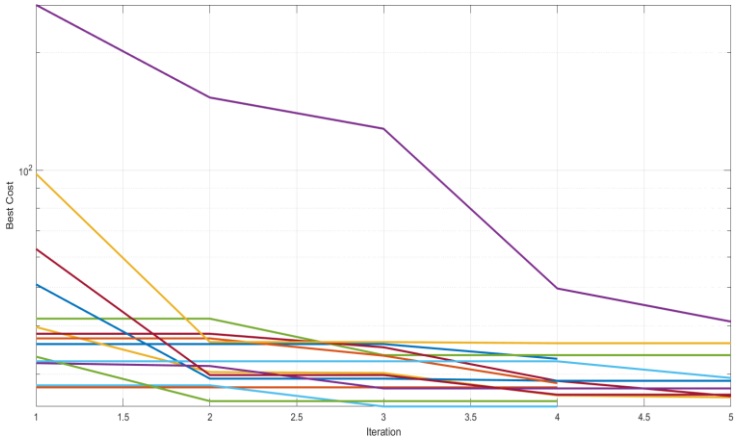


Fig. 20: Grafico dell'evoluzione IAE ottenuto con l'algoritmo PSO

Successivamente, si sono realizzate simulazioni più lunghe in modo tale da poter realizzare un confronto estensivo tra le meta-euristiche. Per queste nuove simulazioni, sono state eseguite, sempre a parità di condizioni, 2 simulazioni per ogni algoritmo, scegliendo ancora di inserire un time-out, ma in questo caso di 3600 secondi (1 ora) sempre per avere un criterio per il raffronto.

Le condizioni iniziali utilizzate per l'esecuzione degli algoritmi, sono:

- Tempo massimo di esecuzione (time-out) pari a 3600 secondi;
- Numero massimo di iterazioni pari a 200;
- Numerosità della popolazione degli agenti pari a 15, per l'Artificial Bee Colony (15 operaie e 15 spettatrici);
- Numerosità della popolazione degli agenti pari a 30, per i restanti algoritmi;
- Lower bound per le variabili decisionali pari a 0.0001;
- Upper bound per le variabili decisionali pari a 100.

Le migliori soluzioni trovate da ogni algoritmo, con IAE scelto come funzione per l'ottimizzazione, sono:

ABC	PSO	GWO	FOA	CBO
21.26	16.16	24.16	26.49	21.16

A differenza dei risultati precedenti, con un time-out di 3600 secondi e aumentando il numero di agenti intelligenti della popolazione, possiamo notare che è il Particle Swarm

Optimization (PSO) l'algoritmo meta-euristico che restituisce il valore di gran lunga migliore a parità di condizioni ed è stato ottenuto a termine di 54 iterazioni a fronte di un numero massimo di 200 iterazioni, seguito subito dopo dal CBO dopo 56 iterazioni, l'ABC dopo 70 iterazioni, il GWO dopo 60 iterazioni ed infine il FOA dopo 98 iterazioni.

Anche per il confronto estensivo, al fine di avere due ulteriori parametri per il confronto delle meta-euristiche sono state calcolate media e varianza delle soluzioni ottenute da ogni algoritmo. I risultati ottenuti sono i seguenti:

	IAE
ABC	Media: 21.21 Varianza: 0.002499
PSO	Media: 16.48 Varianza: 0.1024000
GWO	Media: 25.625 Varianza: 2.146225
FOA	Media: 30.045 Varianza: 12.638025
CBO	Media: 23.655 Varianza: 6.225025

Man a mano che i test venivano eseguiti, si è reso sempre più evidente che l'algoritmo di ottimizzazione che forniva valori soddisfacenti era il PSO, e di conseguenza ci si è concentrati su di esso.

4.3 Particle Swarm Optimization modificato

Visti i promettenti risultati ottenuti nelle simulazioni, si è deciso di concentrarsi sull'algoritmo Particle Swarm Optimization, Sono state testate varie versioni di tale algoritmo, ma la versione che ha fornito i risultati più soddisfacenti è stata la versione logaritmica.

Per realizzare le simulazioni, sono state implementate le seguenti modifiche:

- Aumento del peso d'inerzia da $\omega = 1$ a $\omega = 1.2$, in modo tale da avere un maggior controllo sull'esplorazione e sullo sfruttamento;

- Definizione logaritmica dei bound delle variabili del problema;
- Modifica dei coefficienti dell'apprendimento locale e globale di ogni agente intelligente, da $C_1 = 1.5$ e $C_2 = 2.0$ a $C_1 = C_2 = 2.0$;
- Aumento del peso che va a limitare la velocità massima, pari a 0.5.

Per quanto riguarda la definizione dei bound in scala logaritmica, è stata implementata come segue:

```
% Definizione dell'Upper bound e del Lower bound delle variabili
VarMin = log10(1e-1);
VarMax = log10(100);
-----
% Randomizzazione all'interno dei bound
particle(i).Position = unifrnd(VarMin,VarMax,VarSize);
-----
% Rafforzamento vincoli di posizione
particle(i).Position = max(particle(i).Position,VarMin);
particle(i).Position = min(particle(i).Position,VarMax);
-----
% Calcolo della funzione di fitness
alpha0 = particle(i).Position(:,1);
alpha1 = particle(i).Position(:,2);
particle(i).Cost = CostFunction(10.^alpha0',10.^alpha1');
```

Questo accorgimento permette all'algoritmo di esplorare con uguale probabilità ogni ordine di grandezza dei parametri di controllo. Nel caso di scala non logaritmica, infatti, l'esplorazione randomica è centrata sulla media dell'intervallo di esplorazione, quindi di fatto su un ordine di grandezza elevato dei parametri.

Di seguito sono elencate le condizioni iniziali utilizzate:

- Tempo massimo di esecuzione (time-out) pari a 3600 secondi;
- Numero massimo di iterazioni pari a 200;
- Numerosità della popolazione degli agenti pari a 30;
- Il peso di inerzia posto pari a $\omega = 1.2$;
- Lower bound per le variabili decisionali pari a 0.0001;
- Upper bound per le variabili decisionali pari a 100.

Per questa simulazione si è optato di usare l'ISE come indicatore, che include gli errori quadratici sia in posizione sia in assetto.

4.3.1 Prima Simulazione del PSO logaritmico

La prima simulazione ha fornito come parametri di $alpha0$ e $alpha1$ e come ISE finale

```
####################
####################

alpha0 -0.38 -1.00 -0.05 1.98 1.82 1.01
alpha1 0.61 0.70 -0.72 0.93 0.82 -0.66
ISE    2.11

####################
####################
```

Dove

$$alpha0 = log10[\alpha_{0,x}, \alpha_{0,y}, \alpha_{0,1}, \alpha_{0,2}, \alpha_{0,3}, \alpha_{0,4}]$$
$$alpha1 = log10[\alpha_{1,x}, \alpha_{1,y}, \alpha_{1,1}, \alpha_{1,2}, \alpha_{1,3}, \alpha_{1,4}]$$

Di seguito sono riportati i grafici degli andamenti delle sei variabili, x, y, z, φ, θ e ψ, che permettono di governare il drone con i relativi riferimenti ideali.

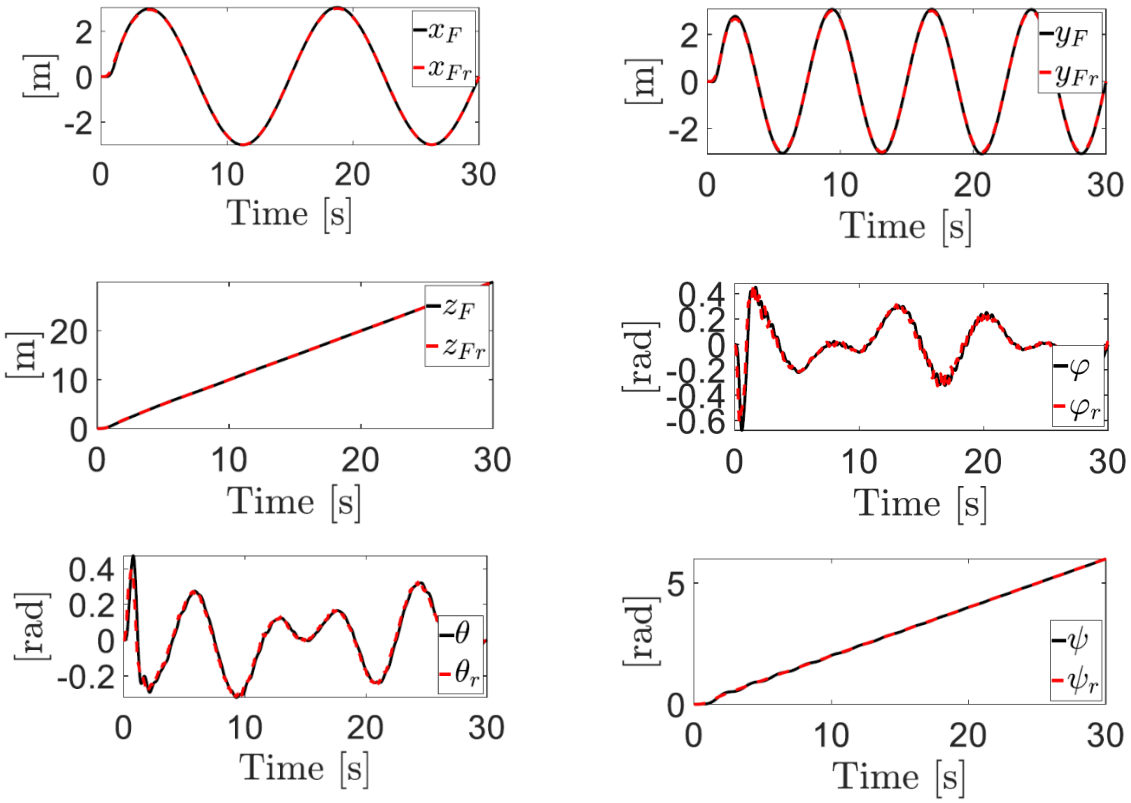


Fig. 21: Grafici degli andamenti delle sei variabili, x, y, z, φ, θ e ψ con i relativi riferimento della prima simulazione PSO logaritmico.

Tali grafici mostrano come gli andamenti simulati di tutti e sei i parametri per il controllo del drone rispettano il riferimento di posizione e orientamento imposto. Si nota, negli angoli di pitch e roll, una leggera oscillazione sia del riferimento che della variabile controllata. L'oscillazione del riferimento deriva dalla struttura di controllo inner-outer che, in presenza di rumore di misura, richiede correzioni successive degli angoli di pitch e roll.

L'ultimo grafico descrive, invece, la traiettoria ideale dall'APR, in rosso, e la traiettoria simulata, in nero. Si può notare che le due traiettorie sono sovrapponibili, infatti, la traiettoria simulata del drone insegue quasi perfettamente la traiettoria ideale, potendo concludere così che i risultati ottenuti sono più che promettenti e soddisfacenti.

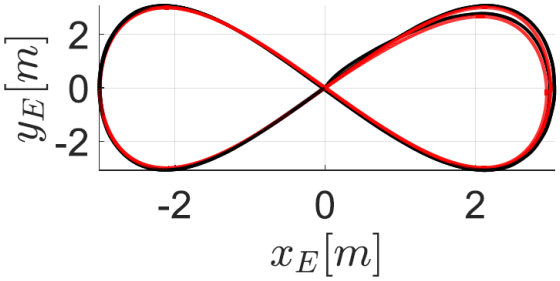


Fig. 22: Grafico della traiettoria simulata dell'esarotore sovrapposta alla traiettoria desiderata

4.6.2 Seconda simulazione del PSO logaritmico

La seconda simulazione ha invece fornito come parametri e come ISE finale:

```
######################
######################

alpha0 0.03 0.78 -0.91 2.00 2.00 1.95
alpha1 0.73 0.85 1.92 1.07 1.08 0.29
ISE    1.96

######################
######################
```

Si nota che l'ISE risulta leggermente migliorato, a fronte di parametri alpha0 e alpha1, in scala logaritmica, tendenzialmente più alti. Di seguito sono riportati i grafici che descrivono gli andamenti dei sei parametri che permettono di governare il drone, con i relativi riferimenti ideali.

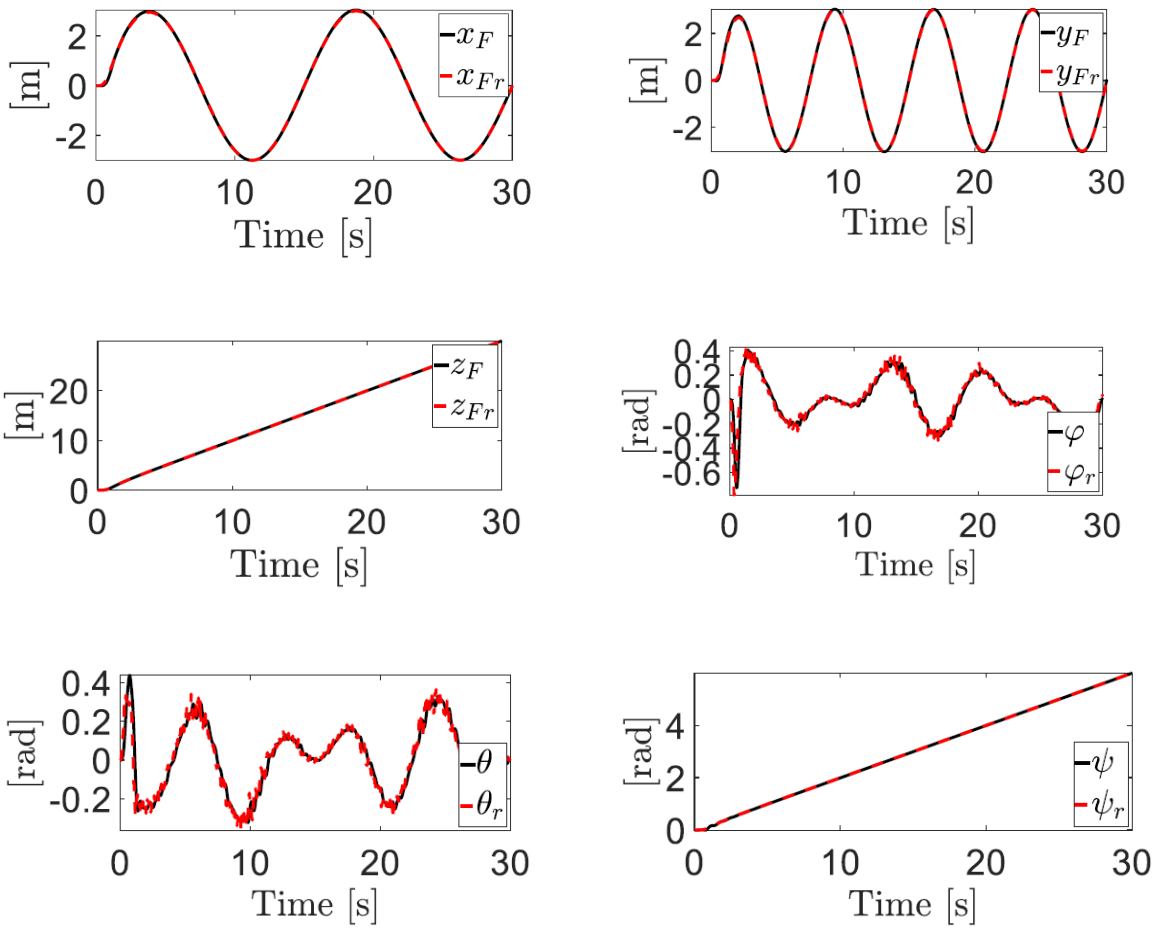


Fig. 23: Grafici degli andamenti delle sei variabili, x, y, z, φ, θ e ψ con i relativi riferimento della prima simulazione PSO logaritmico

Anche in questo caso, si nota una leggera oscillazione del riferimento degli angoli di pitch e roll. Le oscillazioni sono maggiori rispetto al caso precedente, perché i parametri di controllo sono più grandi, quindi richiedono delle correzioni più spinte.

Come detto precedentemente, l'ultimo grafico descrive, invece, la traiettoria ideale dall'APR, in rosso, e la traiettoria simulata, in nero. Il grafico è molto simile al caso precedente. Infatti, i miglioramenti di ISE sono marginali e quindi non sono visibili dal grafico di traiettoria.

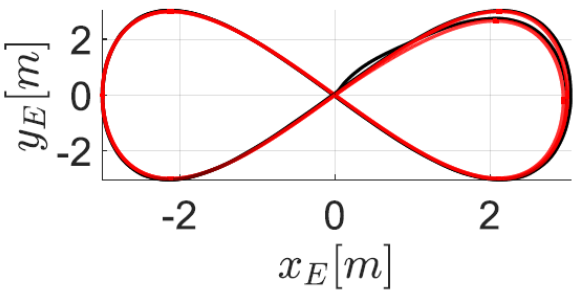


Fig. 24: Grafico della traiettoria simulata dell'esarotore sovrapposta alla traiettoria desiderata

Capitolo 5

Conclusioni e sviluppi futuri

Il seguente elaborato di tesi aveva come obiettivo la definizione di un metodo efficace di taratura dei parametri di un drone esarotore, che permettesse l'inseguimento della traiettoria desiderata. Per perseguire tale obiettivo, sono stati utilizzati vari algoritmi meta-euristici, appartenenti alla categoria della Swarm Intelligence. A tale scopo, il problema di taratura è stato riscritto come un problema di ottimizzazione, ovvero è stato trattato come un problema di minimizzazione di un funzionale, ricercando in un determinato spazio delle soluzioni.

Dopo l'introduzione, dove è stata presentata una panoramica generale sullo sviluppo dei droni nel corso degli anni, i differenti ambiti in cui oggi essi posso essere utilizzati ed infine la descrizione dell'obiettivo di questa tesi, nel Capitolo 2 sono stati illustrati i vari paradigmi di ottimizzazione impiegati allo scopo definito precedentemente, mettendo in evidenza le loro particolarità e il loro funzionamento. Mentre nel capitolo 3, è stato descritto l'esarotore preso come caso di studio, fornendone il modello matematico, lo schema e le leggi di controllo utilizzati e gli indici adoperati. Nel capitolo 4, infine, sono stati riportati e descritti i risultati finali delle varie simulazioni realizzate, con annesse osservazioni.

Durante l'esecuzione delle simulazioni, osservando i migliori risultati ottenuti rispetto agli algoritmi, si è deciso di concentrarsi soltanto su uno dei paradigmi esposti, ovvero il PSO, per il quale sono state realizzate delle modifiche nel suo funzionamento, come per esempio la modifica del peso d'inerzia da $\omega = 1$ a $\omega = 1.2$, in modo tale da avere un maggior controllo sull'esplorazione e sullo sfruttamento, la definizione logaritmica dei bound delle variabili del problema, la modifica dei coefficienti dell'apprendimento locale e globale di

ogni agente intelligente, da $C_1 = 1.5$ e $C_2 = 2.0$ a $C_1 = C_2 = 2.0$ ed infine l'aumento del peso che va a limitare la velocità massima.
Grazie a questi accorgimenti, l'obiettivo prefissato di questa tesi si considera raggiunto a valle delle validazioni e dei test effettuati e presentati, potendo concludere che i risultati ottenuti dalle simulazioni sono soddisfacenti, in quanto la traiettoria simulata dell'esarotore insegue quasi perfettamente la traiettoria ideale.

Infine, possibili sviluppi futuri possono riguardare ulteriori modifiche al codice MATLAB dei paradigmi utilizzati e validare il metodo di taratura con altre leggi di controllo, con altri modelli di drone, al variare della traiettoria di riferimento desiderata ed infine valutare il controllore tarato con prove sperimentali.

Bibliografia

[1] https://it.wikipedia.org/wiki/Aeromobile_a_pilotaggio_remoto

[2] https://www.puntodroni.com/invenzione-drone-storia-significato/

[3] https://enterprise-insights.dji.com/it/blog/4-modi-con-cui-i-droni-combattono-gli-incendi-boschivi

[4] https://www.wired.it/economia/consumi/2019/06/06/amazon-droni-consegne/?refresh_ce=

[5] https://www.rgdroni.it/blog/fotogrammetria-ricostruzione-3d-drone-ad-oggi-cosa-possiamo-ottenere/

[6] https://www.ilmiodrone.it/primo-soccorso-arriva-il-drone-ambulanza/

[7] https://sites.math.washington.edu/~burke/crs/515/notes/nt_1.pdf

[8] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, "Grey Wolf Optimizer", Advances in Engineering Software 69 (2014) 46–61

[9] Dervis Karaboga, Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) Algorithm", 13 April 2007 Springer Science+Business Media B.V. 2007

[10] Riccardo Poli - James Kennedy - Tim Blackwell, "Particle swarm optimization - An overview", August 2007, Springer Science + Business Media, LLC 2007

[11] Wen-Tsao Pan, "A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example", Knowledge-Based Systems 26 (2012) 69–74

[12] A. Kaveh, V.R. Mahdavi "Colliding bodies optimization: A novel meta-heuristic method", Computers and Structures 139 (2014) 18–27

[13] Yarpiz (2021). Artificial Bee Colony (ABC) in MATLAB (https://www.mathworks.com/matlabcentral/fileexchange/52966-artificial-bee-colony-abc-in-matlab), MATLAB Central File Exchange. Retrieved October 11, 2021

[14] Mostapha Kalami Heris, Particle Swarm Optimization in MATLAB (URL: https://yarpiz.com/50/ypea102-particle-swarm-optimization), Yarpiz, 2015

[15] Stephen Zhao (2021). FOA (https://www.mathworks.com/matlabcentral/fileexchange/67796-foa), MATLAB Central File Exchange

[16] https://link.springer.com/content/pdf/bbm%3A978-3-319-19659-6%2F1.pdf

[17] Seyedali Mirjalili (2021). Grey Wolf Optimizer (GWO) (https://www.mathworks.com/matlabcentral/fileexchange/44974-grey-wolf-optimizer-gwo), MATLAB Central File Exchange. Retrieved October 8, 2021.

[19] A. Baldini, R. Felicetti, A. Freddi, S. Longhi and A. Monteriù, "Disturbance Observer Based Fault Tolerant Control for Tilted Hexarotors," 2021 International Conference on Unmanned Aircraft Systems (ICUAS), 2021, pp. 20-27, doi: 10.1109/ICUAS51884.2021.9476843

[20] https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf

Elenco delle figure

Fig. 1: http://machinesofleonardodavinci.blogspot.com/2017/02/helicoptero-helicopter.html

Fig. 2: Giones, Ferran and Alexander Brem. "From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry." Business Horizons 60.6 (2017): 875-884

Figs. 3-7: Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, “Grey Wolf Optimizer”, Advances in Engineering Software 69 (2014) 46–61

Fig. 8: Wen-Tsao Pan, “A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example”, Knowledge-Based Systems 26 (2012) 69–74

Figs. 9-10: A. Kaveh, V.R. Mahdavi, “Colliding bodies optimization: A novel meta-heuristic method”, Computers and Structures 139 (2014) 18–27

Fig. 12: Michieletto, Giulia, Markus Ryll, and Antonio Franchi. "Fundamental actuation properties of multirotors: Force–moment decoupling and fail–safe robustness." IEEE Transactions on Robotics 34.3 (2018): 702-715

Fig. 13: A. Baldini, R. Felicetti, A. Freddi, S. Longhi and A. Monteriù, "Disturbance Observer Based Fault Tolerant Control for Tilted Hexarotors," 2021 International Conference on Unmanned Aircraft Systems (ICUAS), 2021, pp. 20-27, doi: 10.1109/ICUAS51884.2021.9476843.

Fig. 14: https://www.sfu.ca/~ssurjano/spheref.html

Fig. 15: https://www.sfu.ca/~ssurjano/dejong5.html