UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Gestionale

ALGORITMI DI SWARM INTELLIGENCE PER LA REALIZZAZIONE DEL DIGITAL TWIN DI UN EIETTORE

SWARM INTELLIGENCE ALGORITHMS FOR THE EJECTOR DIGITAL TWIN DESIGN

Relatore: Chiar.mo

Prof. **Maurizio Bevilacqua**

Tesi di Laurea di:

Jacopo Venanzi

Correlatore: Chiar.mo

Prof. **Giovanni Mazzuto**

A.A. **2019** / **2020**

Alla mia famiglia,

ai miei amici.

Indice

Introduzione

Al giorno d'oggi il processo di digitalizzazione sta coinvolgendo tutte le varie realtà aziendali. A promuovere tale evoluzione, hanno contribuito e continuano a farlo in maniera determinante, la quarta rivoluzione industriale, anche nota come "Industry 4.0" e le nuove tecnologie introdotte da quest'ultima. L'integrazione di sistemi fisici con nuovi sistemi digitali, l'utilizzo di elementi intelligenti, quali sensori ed attuatori interconnessi e la capacità di elaborare, anche in tempo reale, le informazioni scambiate, consentono un miglioramento netto dell'esecuzione di vari processi operativi. In questo contesto, si introduce il paradigma "Digital Twin", che ha l'obiettivo di concretizzare tutto quello precedentemente riportato. La creazione di gemelli digitali di macchinari, processi, servizi e l'interfacciamento con le rispettive controparti fisiche, garantiscono infatti numerosi vantaggi, tra cui la possibilità di fare previsioni su comportamenti futuri o su eventuali problematiche che potrebbero presentarsi sull' "elemento" reale considerato. L'obiettivo di questo progetto di tesi è quello di individuare ed implementare una corretta tipologia di algoritmi (che risulteranno essere quelli basati sulla Swarm Intelligence) al fine di effettuare l'identificazione parametrica dell'eiettore presente nell'impianto di miscelazione bifase del Dipartimento di Ingegneria Industriale e Scienze Matematiche (DIISM). Lo scopo ultimo infatti, è quello di creare un modello digitale di tale eiettore che sia in grado di rappresentare al meglio il funzionamento dello stesso. Il modello digitale ottenuto, in futuro verrà poi integrato insieme a tutti gli altri modelli digitali dei vari componenti dell'impianto, al fine di creare un Digital Twin dello stesso. Quest'ultimo permetterà di simulare diverse situazioni di funzionamento dell'intero impianto e di prevedere eventuali situazioni di pericolo e/o anomalie (Anomaly Detection), al fine di salvaguardare l'impianto stesso da possibili danneggiamenti e prevenire i rischi per gli operatori.

Capitolo 1

Eiettore e Impianto di miscelazione bifase

In questo primo capitolo verrà presentato l'elemento chiave su cui è basato questo progetto di tesi: l'eiettore. Verranno analizzati i campi applicativi in cui è presente e in particolare il suo funzionamento, facendo dei richiami al noto effetto Venturi. Successivamente sarà esaminato anche l'impianto, presente nel Dipartimento di Ingegneria Industriale e Scienze Matematiche dell'Università Politecnica delle Marche, in cui è collocato l'eiettore. Infine verranno analizzate varie tecniche di modellazione, al fine di individuare le migliori, che consentiranno quindi una corretta identificazione parametrica dell'eiettore.

1.1 Definizione di eiettore

Con il termine eiettore si intende un'apparecchiatura, senza organi meccanici in movimento, in cui l'energia cinetica di un fluido in moto, detto fluido primario o motore, viene trasferita, completamente o in parte mediante miscelamento, ad un altro fluido, detto fluido secondario o aspirato. Una volta avvenuto il miscelamento, l'energia cinetica posseduta dalla miscela dei due fluidi può essere convertita in un aumento di pressione, tramite il passaggio in un diffusore. Grazie a questo meccanismo il fluido aspirato si porta ad una pressione finale superiore a quella dell'ambiente da cui viene prelevato. Si ottiene dunque una compressione dello stesso, a spese di una riduzione di energia del fluido motore.

Tale dispositivo può prevedere in alimentazione l'impiego di fluidi comprimibili o di fluidi incomprimibili, potendo lavorare anche con fluidi di natura diversa. Questo componente inoltre può anche essere definito "Pompa a Getto" (Jet Pump). La pompa a getto, infatti, classificata anche come macchina operatrice statica, è costituita da canali fissi senza organi meccanici rotanti.

Un esempio rappresentativo dell'eiettore è mostrato in *fig.1*.

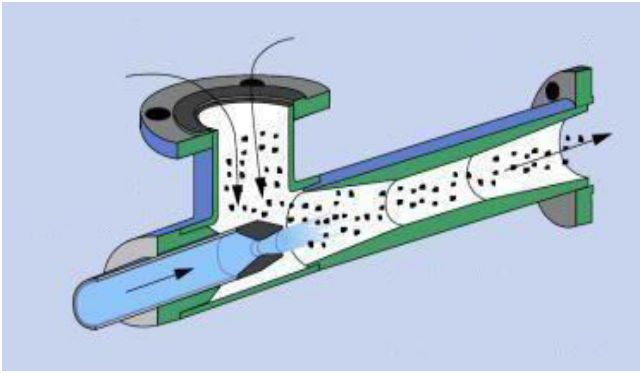


fig.1

1.2 Cenni Storici, Applicazioni e Tipologie

Uno dei primi prototipi dell'eiettore, che ad oggi risulta ancora essere utilizzato in diversi ambiti industriali, fu inventato nel 1858 da Henry Giffard. Tale prototipo (*fig.2*) all'epoca fu utilizzato per risolvere il problema dell'alimentazione dell'acqua ai serbatoi degli impianti a vapore.

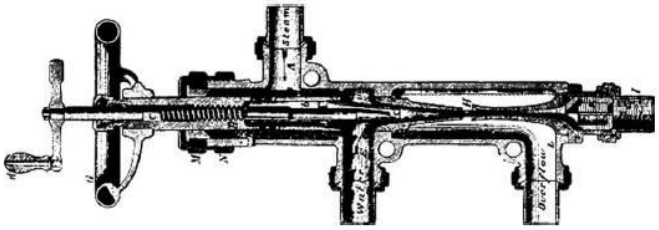


fig.2

Fino a quel momento infatti, l'energia necessaria per la movimentazione dell'acqua alle caldaie dell'impianto, veniva fornita da pompe meccaniche. Queste ultime oltre a non essere molto affidabili, erano a loro volta alimentate da motori funzionanti a vapore. Tutto ciò non garantiva l'alimentazione a impianto fermo, in quanto l'energia motrice veniva a mancare e quindi era impedito un adeguato apporto d'acqua costante all'impianto. (Kranakis 1982)

L'invenzione di Giffard, come accennato precedentemente, permise di risolvere questa problematica, ossia consentì la movimentazione di acqua (in forma liquida) a bassa pressione, sfruttando l'energia del vapore a più alta pressione. L'eiettore permise quindi di alimentare il circuito in modo costante anche quando le tradizionali pompe non erano in grado di funzionare.

Il dispositivo ideato da Giffard, insieme agli altri primi prototipi di eiettori, era dotato di un ugello principale a forma convergente. Nel 1869 invece, per la prima volta, Schau introdusse l'eiettore con ugello convergente – divergente, sfruttando gli studi effettuati da Gustaf de Laval. Questo ultimo successivamente, nel 1890 riuscì a sperimentare e brevettare un altro tipo di ugello: l'ugello supersonico di tipo convergente – divergente. Tale dispositivo, ancora oggi utilizzato, sfrutta l'azione di un fluido supersonico e di uno subsonico. In *fig.3* è rappresentato uno schema dell'ugello convergente – divergente, conosciuto anche come "ugello De Laval" dove viene anche riportata la direzione del flusso del fluido interno.

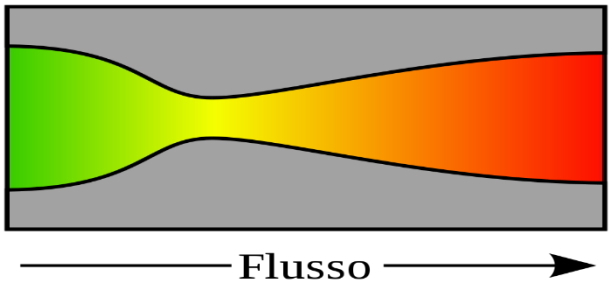


fig.3

In seguito alle loro varie invenzioni, gli eiettori vennero ampiamente testati al fine di essere applicati in diversi settori industriali. Tali dispositivi furono maggiormente utilizzati, in particolare, nel campo della refrigerazione. (Elbel 2011)

Nel 1910 Maurice Leblanc, introdusse nei cicli a vapore, l'eiettore a getto di vapore. Questo dispositivo permetteva di produrre un effetto frigorifero impiegando fonti di basso grado energetico. Da quel momento i sistemi di refrigerazione a getto di vapore furono largamente utilizzati, come ad esempio per il condizionamento dell'aria, in grandi edifici o nei vagoni dei treni. Dopo circa 20 anni, fu sviluppato l'eiettore bifase. Tale scoperta è da attribuire a Norman H. Gay, che lo brevettò nel 1931 al fine di migliorare le performance dei sistemi di refrigerazione, riducendo le perdite energetiche dovute all'uso delle valvole di espansione. (Gay 1931) Tale tecnologia viene tutt'ora utilizzata in svariati ambiti industriali, in particolare la sua applicazione nei cicli operanti ad anidride carbonica risulta essere molto vantaggiosa.

La versatilità degli eiettori ne permette l'utilizzo in un ampio campo di applicazioni, dove prevalgono necessità come "semplicità costruttiva", "compattezza", "affidabilità" e "sicurezza".

(Chunnanond e Aphornratana 2004) e (Tashtoush, Al-Nimr, e Khasawneh 2019) hanno presentato un'analisi della letteratura inerente all'applicazione degli eiettori in diversi settori. Di seguito ne sono riportati alcuni esempi:

- eiettori usati per la refrigerazione dei cibi insieme all'utilizzo di sistemi frigoriferi tradizionali;
- eiettori usati per il pompaggio dell'acqua e del vapore nei sistemi di emergenza delle centrali nucleari per garantire il raffreddamento del nocciolo anche in situazioni di emergenza;
- eiettori usati nell'industria chimica per il pompaggio di sostanze pericolose;
- eiettori usati nei sistemi di propulsione aeronautica per l'aumento della spinta e la diminuzione della temperatura dei gas di scarico;
- eiettori usati nell'industria cartaria per il recupero e la ri-circolazione del vapore nei cilindri essiccatori;
- eiettori usati nelle centrali elettriche per l'evacuazione di condensatori a valle di turbine;
- eiettori usati nelle costruzioni navali per lo svuotamento di serbatoi o spazi di zavorra.

In base al tipo di applicazione in cui l'eiettore viene utilizzato, cambiano i tipi di fluidi impiegati. In merito a questo, 4 tipologie principali di eiettore possono essere classificate, come mostrato in *tab.1* (Elbel 2011):

Tipologia Eiettore	Fluido Motore	Fluido Aspirato	Fluido in uscita
a Getto di Vapore	vapore	vapore	vapore
a Getto Liquido	liquido	liquido	liquido
a Condensazione	vapore	liquido	liquido
Bifase	liquido	vapore	bifase (liquido-vapore)

tab.1

1.3 Principio di funzionamento dell'eiettore

Il funzionamento dell'eiettore è basato sull'effetto Venturi sviluppato da un ugello convergente - divergente per convertire l'energia data da un fluido motore ad elevata pressione in un aumento di velocità dello stesso. Questo fenomeno genera una zona di depressione che trascina quindi, all'interno dell'eiettore un secondo fluido (fluido aspirato) presente a pressione inferiore rispetto a quella del fluido motore. Superata la zona di convergenza, i due fluidi si mescolano

nella zona di miscelazione e successivamente il fluido miscelato tende ad espandersi in corrispondenza di un diffusore dove la sua velocità si riduce, comportando un aumento della pressione.

1.3.1 Effetto Venturi

L'effetto Venturi, scoperto e studiato da Giovanni Battista Venturi, è un fenomeno fisico per cui la pressione di una corrente fluida aumenta con il diminuire della velocità. Considerando un generico condotto (*fig.4*) avente una diminuzione della sezione al suo interno e percorso da un fluido a densità costante (incomprimibile), per l'equazione di conservazione applicata alla fluidodinamica è possibile affermare che la portata entrante nella sezione maggiore A_1, è uguale a quella passante nella sezione minore A_2.

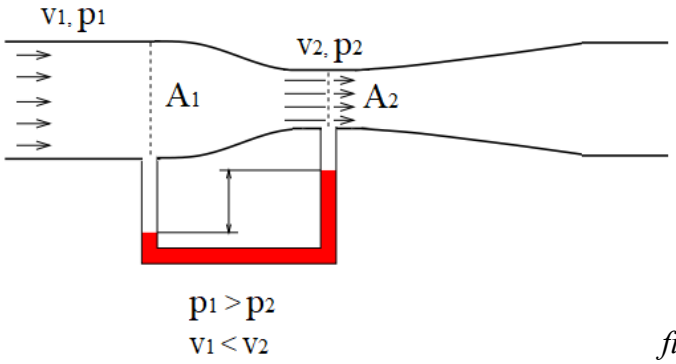


fig.4

Dato che la portata Q può essere espressa come prodotto tra velocità del fluido v e la sezione in cui passa A, si ha:

$$Q = v_1 * A_1 = v_2 * A_2$$

Da questa relazione si deduce che ad una diminuzione della sezione (A_2) corrisponde un aumento della velocità (v_2).

Applicando inoltre la famosa equazione di Bernoulli tra le due sezioni del condotto:

$$\rho g h_1 + p_1 + \frac{1}{2}\rho v_1^2 = \rho g h_2 + p_2 + \frac{1}{2}\rho v_2^2$$

ed ipotizzando che non sussista una differenza di quota (altezza h) tra le due sezioni, si può giungere alla seguente espressione:

$$p + \frac{1}{2}\rho v^2 = \ costante$$

Dove:

- p = pressione del fluido [Pa]
- ρ = densità del fluido [kg/m^3]
- v = velocità del deflusso [m/s]

Si ottiene quindi una correlazione tra la pressione e la velocità in una data sezione: all'aumentare della velocità del fluido, si crea necessariamente una diminuzione della pressione interna dello stesso, tale da mantenere la loro somma *costante*. Nella strozzatura quindi dato che la velocità (v_2) aumenta, la pressione (p_2) diminuisce.

In conclusione si hanno quindi le seguenti relazioni:

$$\begin{cases} Q = v_1 * A_1 = v_2 * A_2 \\ p_1 - p_2 = \frac{1}{2}\rho(v_2^2 - v_1^2) \end{cases}$$

L'Effetto Venturi viene anche definito paradosso idrodinamico, poiché è comunemente pensabile che, all'interno di strozzature, la pressione di un fluido tendi ad aumentare, tuttavia, come visto precedentemente, accade il contrario.

1.3.2 Analisi del funzionamento

L'eiettore, come si può vedere in *fig.5*, è costituito da varie parti fondamentali:

- bocchello di aspirazione → permette l'ingresso del fluido aspirato (secondario);
- bocchello del fluido motore → permette l'ingresso del fluido motore (primario);
- ugello convergente → espande ed accelera adeguatamente il fluido motore;
- zona/camera di aspirazione;
- zona/camera di miscelazione;
- zona/camera di diffusione (o diffusore).

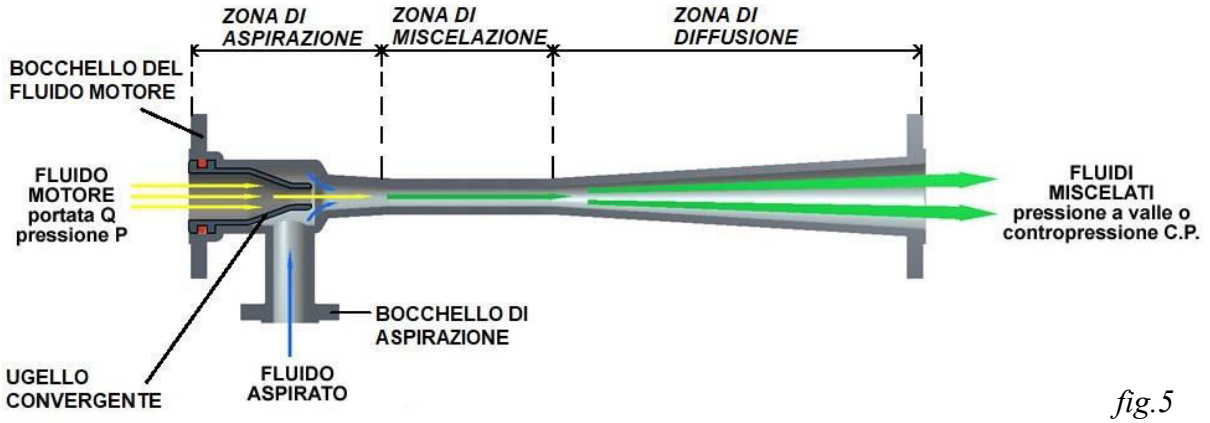


fig.5

La presenza dell'ugello e la geometria dell'eiettore stesso fanno sì che in ciascuna zona siano identificati diversi fenomeni. Tali fenomeni permettono di ottenere il processo di miscelazione tra i due fluidi di alta e bassa pressione, che miscelandosi si portano ad una pressione intermedia (contropressione).

Analizzando il funzionamento dell'eiettore si possono individuare 4 fasi:

1) <u>Fase del fluido motore:</u>

   Il fluido motore o primario, entra all'interno dell'ugello dove viene accelerato grazie alla particolare forma convergente di questo ultimo. Allo stesso tempo, in accordo con l'effetto Venturi e con il principio di Bernoulli, il fluido motore diminuisce la propria pressione. L'energia statica di pressione viene convertita in energia cinetica;

2) <u>Fase del fluido aspirato:</u>

   Il fluido motore, dopo essere stato accelerato nell'ugello, è immesso nella camera di aspirazione di forma convergente dove si è creata una depressione a causa sempre dell'effetto Venturi. Tale depressione permette l'aspirazione del fluido secondario (aspirato), che si trova ad una pressione inferiore del fluido primario;

3) <u>Fase di miscelazione:</u>

   I due fluidi successivamente passano all'interno della zona di miscelamento, caratterizzata da una sezione costante del condotto. Qui i due fluidi si miscelano e scambiano fra loro una certa quantità di energia. Grazie all'elevata quantità di moto del fluido motore, anche il fluido aspirato viene quindi accelerato;

4) Fase di diffusione:

   La miscela formatasi tra i due fluidi entra all'interno del zona di diffusione di forma divergente (la sezione del condotto tende ad aumentare man mano). Il diffusore, in accordo con l'equazione di Bernoulli, permette di ridurre la velocità della miscela e contemporaneamente di aumentarne la pressione. L'energia cinetica è quindi riconvertita in energia statica di pressione. Il fluido miscelato, in uscita all'eiettore, risulta avere una pressione maggiore.

1.4 Pregi e Difetti

Nonostante gli eiettori in genere abbiano una bassa efficienza, solitamente mai superiore al 30% ed indicata come il rapporto tra il lavoro isoentropico di compressione del fluido secondario e il lavoro isoentropico di espansione del fluido primario, la loro semplicità costruttiva e l'assenza di parti mobili al loro interno garantiscono diversi vantaggi sia in termini di affidabilità che in termini economici (Priano 2013). I principali vantaggi derivanti l'uso di eiettori sono i seguenti:

- alta affidabilità costruttiva (non avendo parti mobili al loro interno);
- diminuzione del livello di vibrazioni durante il funzionamento;
- bassi costi di investimento iniziale;
- minimi costi di operabilità, grazie all'assenza di sistemi di lubrificazione (a volte necessari invece con l'uso di compressori);
- riduzione in termini di costi e tempi legati alla manutenzione di tali dispositivi.

1.5 Impianto di Miscelazione Bifase

In questo progetto di tesi, come detto precedentemente, si effettuerà un'identificazione parametrica al fine di realizzare un modello digitale (Digital Twin) dell'eiettore (*fig.6*) applicato all'impianto di miscelazione bifase presente nel DIISM (Dipartimento di Ingegneria Industriale e Scienze Matematiche - UNIVPM).

fig.6

L'impianto di miscelazione acqua-aria è composto principalmente da 3 componenti, ossia un eiettore, una pompa e un serbatoio (o separatore verticale) i quali sono collegati tramite delle tubazioni, che confluiscono poi in una vasca di accumulo dell'acqua che circola nell'impianto stesso. Oltre questi componenti, nell'impianto sono installati vari tipi di sensori di pressione (assoluta – relativa – differenziale) e due sensori di portata volumetrica. Sono presenti inoltre elettrovalvole pneumatiche comandate da alcuni sensori appena descritti e valvole di intercettazione a 2 vie (*fig.7*).

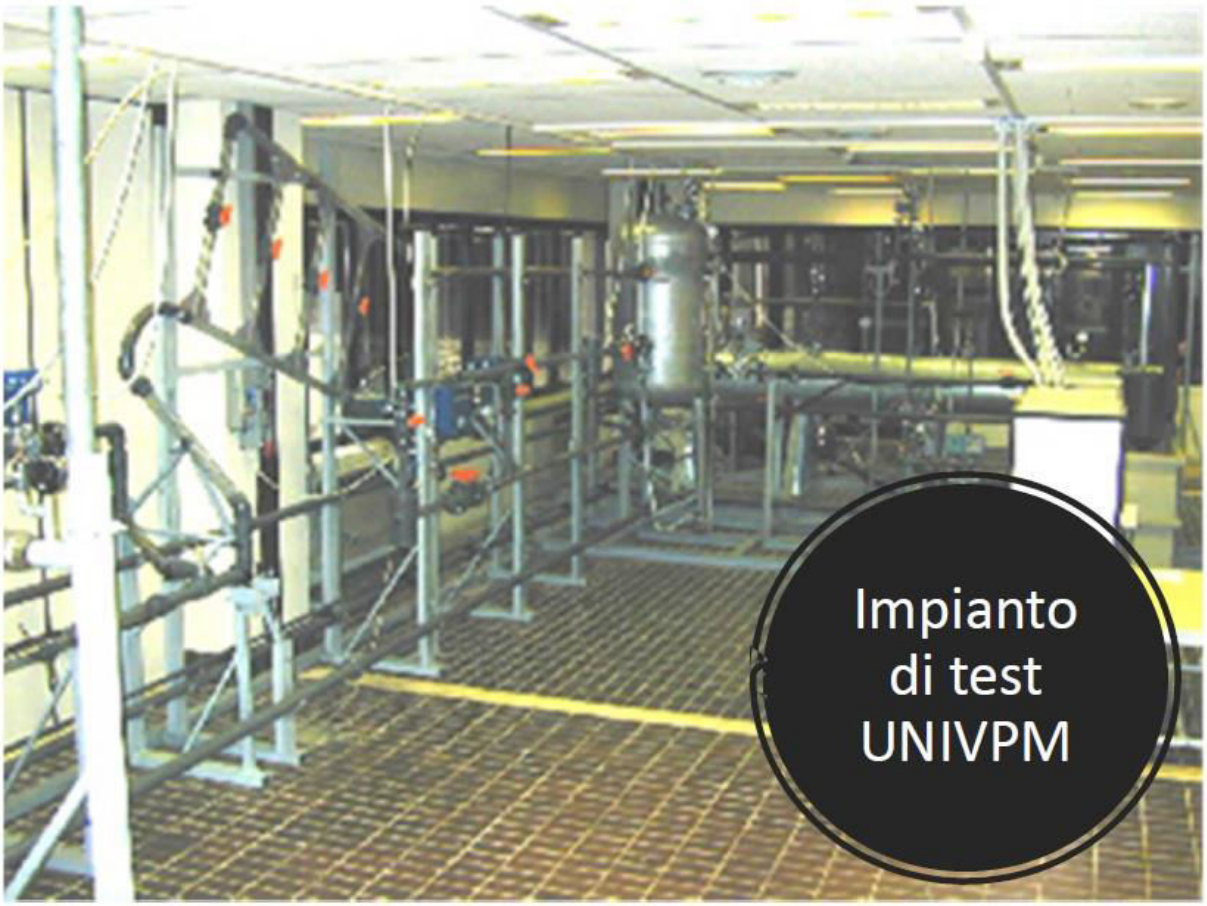


fig.7

In *fig.8* viene riportato lo schema impiantistico, mentre in *fig.9* viene riportata la descrizione dei vari indici presenti nello schema.

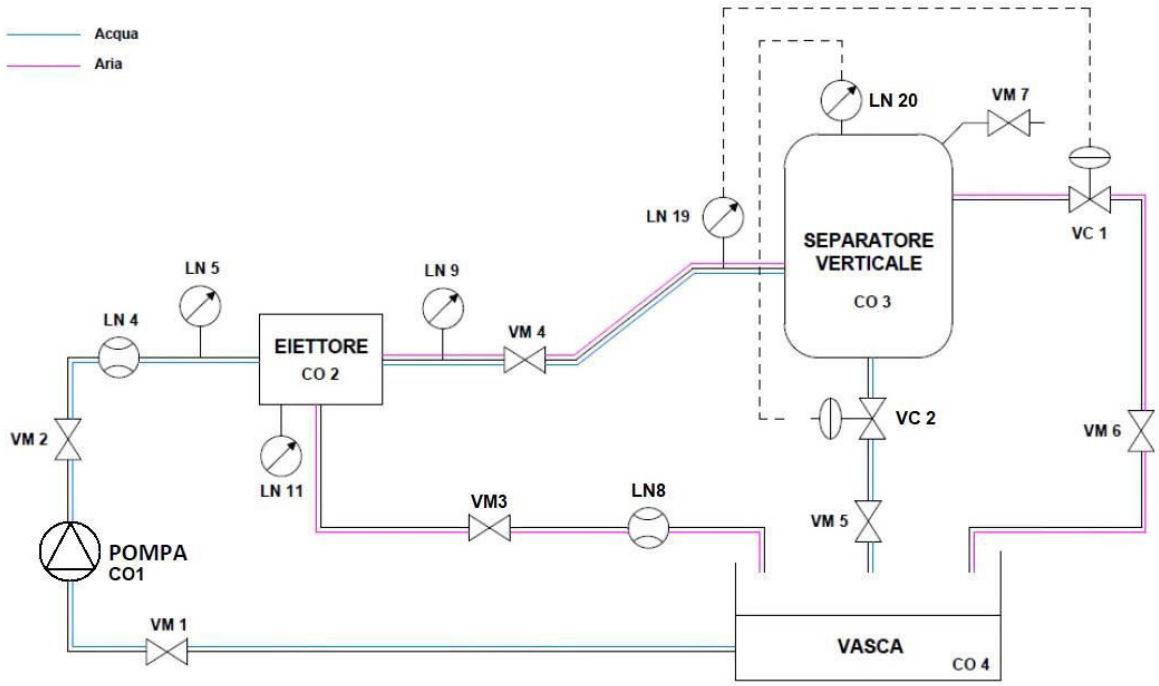


fig.8

Indice	Descrizione
CO1	Pompa Centrifuga monoblocco Vergani 32-201
CO2	Eiettore
CO3	Serbatoio/Separatore verticale Elbi 780-I
CO1	Serbatoio Aperto/Vasca
LN4	Sensore portata Foxboro Magnetic Flowtransmitter
LN5	Sensore pressione assoluta Setra 280E
LN11	Sensore pressione assoluta Setra 280E
LN9	Sensore pressione assoluta Setra 280E
LN19	Sensore pressione relativa Foxboro 841GM-CI1
LN20	Sensore pressione differenziale Foxboro IDP-10
LN8	Sensore portata volumetrica Foxboro Vortez DN 50
VC1	Elettrovalvola pneumatica ECKARDT MB6713
VC2	Elettrovalvola pneumatica ECKARDT MB6713
VM1	Valvola di intercettazione 2 vie FIP DN 50 PN 16
VM2	Valvola di intercettazione 2 vie FIP DN 50 PN 16
VM3	Valvola di intercettazione 2 vie FIP DN 50 PN 16
VM4	Valvola di intercettazione 2 vie FIP DN 50 PN 16
VM5	Valvola di intercettazione 2 vie FIP DN 50 PN 16
VM6	Valvola di intercettazione 2 vie FIP DN 50 PN 16

fig.9

L'impianto era stato inizialmente progettato per simulare un fenomeno che si utilizza nei processi di estrazione petroliferi, cioè quello di sfruttare un giacimento che presenti una pressione più alta di quella del trasporto in linea (del petrolio), al fine di creare un'aspirazione su un giacimento la cui pressione invece non risulta essere sufficientemente alta per il trasporto. Tramite l'utilizzo dell'eiettore si riescono così a sfruttare anche le risorse di pozzi petroliferi non più produttivi (*fig.10*).

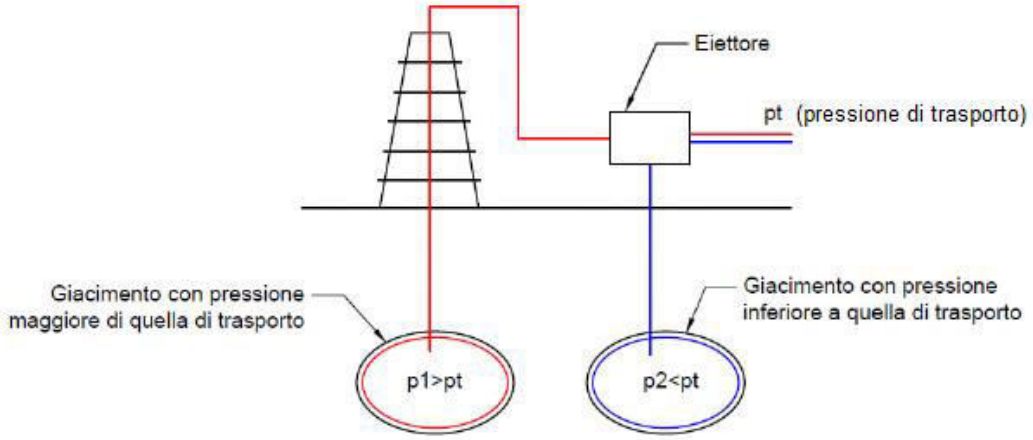


fig.10

Ad oggi, invece, l'impianto viene studiato al fine di creare un prototipo di Digital Twin dello stesso. Tale modello virtuale verrà utilizzato per analizzare possibili scenari di malfunzionamento/avarie e per prevenire l'accadimento di questi ultimi, garantendo quindi un corretto funzionamento operativo dell'impianto.

1.5.1 Funzionamento generale dell'impianto

L'acqua viene mandata dalla vasca di accumulo all'eiettore tramite la pompa che ne alza la pressione. Nell'eiettore viene quindi aspirata aria dall'ambiente esterno a causa del fenomeno di depressione che si crea al suo interno, nella zona in prossimità del convergente. I due fluidi quindi si mescolano nell'eiettore e la miscela acqua/aria che ne esce viene mandata poi al serbatoio (separatore verticale). In tale componente impiantistico sono presenti opportuni sistemi di controllo formati da sensori ed elettrovalvole, che permettono di fissare il valore della pressione e di mantenere costante il livello di acqua presente all'interno del serbatoio. L'acqua in eccesso fluisce attraverso una tubazione dal serbatoio alla vasca di accumulo, mentre l'aria in eccesso viene scaricata all'esterno tramite una valvola.

1.6 Tecniche di modellazione dell'eiettore

Effettuando un'analisi della letteratura si possono riscontare varie tecniche utilizzate per risolvere il problema della modellazione dell'eiettore, ossia tecniche che studiano il comportamento dinamico dello stesso, in particolare il comportamento fluidodinamico dei liquidi/gas che lo attraversano, al fine di prevedere ed ottimizzare il suo funzionamento in diverse condizioni operative. In particolare, nei sistemi di refrigerazione ad eiettore, le prestazioni di tale componente influiscono molto sul funzionamento/andamento generale del sistema; risulta quindi fondamentale controllarle. Queste tecniche possono quindi anche essere usate per "guidare" il funzionamento dell'intero sistema, interpretare i risultati sperimentali e assistere l'ottimizzazione dello stesso.

Uno dei primi approcci alla modellazione dell'eiettore, per la previsione delle prestazioni dello stesso, è stato proposto da Huang et al. nel 1999 (Huang et al. 1999). Gli autori hanno proposto un'analisi 1D, ossia un modello matematico che utilizza un set di equazioni algebriche, equazioni differenziali ed equazioni di conservazione della termodinamica, che caratterizzano il comportamento dell'eiettore (Narimani et al. 2019). Tale metodo è stato utilizzato poi anche da altri autori come Nicolas Galanis e Mikhail Sorin (Galanis e Sorin 2016).

Una delle principali tecniche di modellazione/analisi del comportamento fluidodinamico dell'eiettore, tutt'ora utilizzata, è l' analisi computazionale fluidodinamica (anche detta CFD). Questa tecnica è stata adoperata, sempre per lo scopo sopra descritto, da vari autori: (Kim et al. 2007), (Li e Li 2011), (Zheng, Li, e Qin 2018), (Mazzelli, Giacomelli, e Milazzo 2018). L'analisi CFD è in grado di risolvere complicate equazioni, che caratterizzano il moto dei fluidi, la cui risoluzione analitica risulta invece fattibile solo nel caso in cui si considerano flussi laminari (non turbolenti). Questa tecnica generalmente effettua calcoli/operazioni con un elevato costo computazionale.

(Liu, Wang, e Jia 2017) hanno proposto un ulteriore modello basato su equazioni multi-parametriche in particolare per stabilire il modello termodinamico di un eiettore, mentre (Zhu et al. 2007) hanno introdotto lo Shock Circle Model ed hanno dimostrato che questo ultimo risulta essere più semplice dei metodi di modellazione 1D e inoltre può prevedere con più precisione anche le prestazioni degli eiettori in modalità critiche.

1.6.1 Considerazioni finali

Le metodologie viste sin ora, sono contraddistinte dal fatto che utilizzano calcoli vettoriali ed equazioni differenziali che possono risultare complesse e che a volte implicano, per la risoluzione delle stesse, la conoscenza di determinate grandezze fisiche specifiche dei fluidi, come ad esempio l'entalpia o l'entropia, che nel nostro caso non sono misurabili. Per di più, se si considera l'analisi CFD, tale tecnica oltre a richiedere risorse di calcolo elevate per l'elaborazione, richiede anche la conoscenza di software per l'implementazione dell'algoritmo di modellazione come, ad esempio, ANSYS Fluent, NUMECA e STAR-CD.

Tali complessità, se adottassimo queste tecniche, renderebbero impossibile o comunque sia molto difficile, la realizzazione del modello virtuale dell'eiettore in esame. Un altro aspetto importante da considerare è il fatto che la tecnica di modellazione dovrà fornire un Digital Twin dell'eiettore che successivamente dovrà interfacciarsi e "comunicare" in modo agevole con gli altri Digital Twin dei componenti dell'impianto di miscelazione (pompa e serbatoio) e con i vari sensori presenti, al fine di realizzare un Digital Twin dell'intero sistema. Per ovviare alle problematiche esposte, e rispettare le esigenze di "inrterfacciabilità" in questo progetto di tesi verranno considerate tecniche basate sul machine learning, ossia sull'apprendimento automatico; in particolare si analizzeranno algoritmi di Swarm Intelligence e Reti Neurali Artificiali (vd. Capitolo 3).

Capitolo 2

Digital Twin e Identificazione parametrica

Nel capitolo corrente verranno studiati 2 concetti fondamentali: il Digital Twin e l'identificazione parametrica. In primo luogo verrà esaminata l'evoluzione nella storia del Digital Twin. Di seguito saranno analizzate la struttura classica di un sistema Digital Twin e le funzionalità di questo ultimo. Verranno inquadrati inoltre anche i vantaggi competitivi che tale tipo di tecnologia riesce a garantire. Nelle ultime sezioni sarà introdotta l'identificazione parametrica e verrà spiegato perché l'eiettore, preso in esame, possa essere considerato un "sistema Black-Box".

2.1 Concetto di Digital Twin

In questi ultimi 10 anni, l'evoluzione tecnologica ha accresciuto in maniera sostanziale la dinamicità dei principali mercati e settori industriali. Se prima per testare i prototipi fisici di prodotti, che si desideravano realizzare, si utilizzava il classico metodo "trial and error", ad oggi questa metodologia risulta (per la maggior parte dei casi) inadeguata, poiché la velocità richiesta per lo sviluppo di un nuovo prodotto è elevatissima e i lead time produttivi sono molto ridotti, quindi non sono permesse perdite di tempo. Le aziende desiderano evitare di imbattersi in problemi durante le fasi di produzione, assemblaggio e testing del proprio prodotto. Per far ciò utilizzano tecniche di simulazione, che appunto simulano, in maniera accurata, sia il funzionamento del prodotto stesso, sia l'esecuzione del processo produttivo, formando il cosiddetto "Digital Twin".

Il "gemello digitale", di fatto, è una rappresentazione virtuale di qualcosa di reale, che consente un'interazione semplice, estremamente veloce e flessibile. Un modello che è in grado di riprodurre fedelmente il comportamento di un intero processo o di uno specifico prodotto, in relazione a stimoli provenienti dall'esterno. Tale modello virtuale può essere utilizzato in ottica sperimentale e previsionale per ottimizzare il funzionamento del processo/prodotto, per testare

condizioni operative particolari e fuori standard. I Digital Twin quindi consentono di dimostrare l'impatto di modifiche alla progettazione, scenari di utilizzo e un'infinità di altre variabili, eliminando il ricorso ai prototipi fisici. (Negri, Fumagalli, e Macchi 2017)

L'efficacia di questa rappresentazione virtuale è amplificata dall'utilizzo combinato delle nuove tecnologie, dall'Internet of Things, sino alla realtà aumentata. Tali tecnologie sono in grado di amplificare la capacità del Digital Twin, di rappresentare fedelmente i componenti fisici, il loro stato e le interazioni che ne caratterizzano il funzionamento. (Schluse e Rossmann 2016)

Il massimo beneficio è ottenuto "sincronizzando" il modello fisico (reale) con quello digitale (virtuale), integrando i dati elaborati da questo ultimo, con le rilevazioni effettuate tramite dei sensori montati sugli elementi reali. Il Digital Twin utilizza infatti i dati dei sensori per determinare le prestazioni in tempo reale, le condizioni operative e i cambiamenti nel tempo degli oggetti reali.

Non si parla più, quindi, di soli prodotti/processi intelligenti, ma di un sistema intelligente composto da elementi fisici a cui corrispondono i rispettivi gemelli digitali. Tale sistema permetterà di implementare nuovi modelli in grado di incrementare la competitività dei prodotti/processi in tutte le fasi del ciclo di vita. Sfruttando al meglio questa opportunità, le aziende potranno sviluppare un vero e proprio vantaggio competitivo.

La *fig.11* mostra una rappresentazione schematica di quello che può essere considerato un Digital Twin.

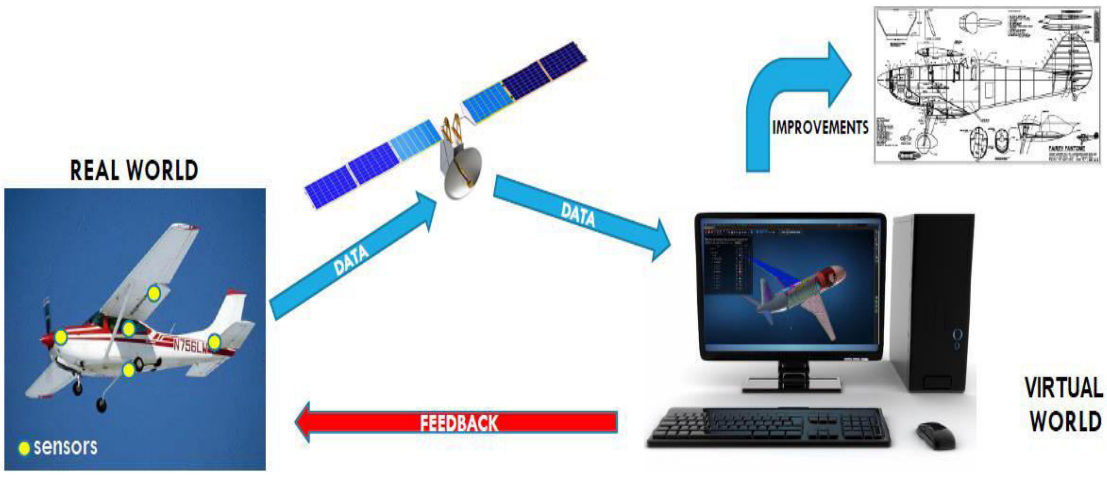


fig.11

2.2 Evoluzione del Digital Twin

2.2.1 Ieri

Il concetto di Digital Twin fu introdotto per la prima volta dall'americano Michael Grieves nel 2002. Grieves descrisse un sistema con il quale si potesse replicare tutte le caratteristiche e le modalità di funzionamento di un prodotto in maniera digitale, distaccandosi quindi dal mondo reale.

Una delle prime concretizzazioni, dell'innovativa idea di Grieves, fu da parte della NASA (Boschert e Rosen 2016). Il problema principale della NASA consisteva nel fatto che doveva lavorare con dei dispositivi che fossero in grado di funzionare correttamente nello spazio, ambiente difficilmente replicabile sulla Terra. Realizzarono quindi una struttura in grado di poter "creare" gli stessi dispositivi in maniera però del tutto virtuale, sui quali poi riuscirono ad effettuare dei test e a simulare le condizioni degli ambienti spaziali. Successivamente interfacciarono questo modello virtuale con il modello reale del dispositivo aerospaziale, al fine di mantenere la copia digitale sempre aggiornata sulle condizioni del prodotto fisico.

2.2.2 Oggi

Con l'avvento delle nuove tecnologie come ad esempio l'IoT (Internet of Things), i Big Data e la nascita dell' Industry 4.0, i sistemi di Digital Twin in questi ultimi anni si sono sviluppati in maniera esponenziale, trovando sempre più impiego in molti settori industriali e non. Molte note imprese hanno deciso di sfruttare tale tecnologia al fine di trarne un vero e proprio vantaggio competitivo, riducendo in maniera sostanziale i costi aziendali. (Saracco 2018)

Ad oggi la NASA, pioniere dei sistemi Digital Twin, utilizza questi ultimi principalmente per sviluppare nuove roadmap spaziali, veicoli ed aerei di nuova generazione.

Dallara, azienda italiana costruttrice di automobili, anziché sviluppare prototipi di scocche e testarli in gallerie del vento e altri laboratori, costruisce dei modelli digitali e su questi effettua tutte le prove e le simulazioni necessarie ad arrivare ad un progetto soddisfacente, velocizzando così tutto il processo di progettazione.

General Electric da alcuni anni ha esteso l'utilizzo del Digital Twin delle sue turbine per effettuare operazioni di manutenzione e controllo. Nelle turbine reali sono presenti diverse tipologie di sensori che in tempo reale inviano dati ai rispettivi modelli virtuali, informandoli sulle condizioni di utilizzo in corso. Tutte le informazioni ricevute sono molto importanti perché consentono da un lato di simulare il comportamento del Digital Twin e dall'altro di rilevare difformità di funzionamento tra il Digital Twin e la turbina reale.

Tesla costruisce un Digital Twin per ogni sua auto elettrica. L'azienda ogni giorno riceve informazioni dalle sue auto (reali) in merito a: dove stanno viaggiando, gli ostacoli identificati lungo il tragitto, il funzionamento del motore, ecc. Questa enorme mole di dati permette di sviluppare una mappa delle strade aggiornata in maniera costante, e di verificare la potenziale presenza di malfunzionamenti strutturali, legati alla progettazione. In pratica il sistema virtuale permette di integrare le operazioni di testing iniziale delle nuove auto con i dati derivanti dall'uso reale, garantendo così un miglioramento continuo del prodotto nelle successive versioni.

Siemens fornisce soluzioni Digital Twin per il design e la prototipazione, ma anche per la produzione e la gestione logistica. Questo insieme di soluzioni permette quindi di digitalizzare intere linee produttive e ridefinire le fasi di prototipazione e testing prodotti. L'uso di intelligenze artificiali, inoltre, migliora la capacità di individuare eventuali anomalie e la capacità di autoapprendimento dei modelli Digital Twin.

2.3 Struttura tipica di un sistema Digital Twin

Il sistema Digital Twin, illustrato precedentemente in *fig.11*, è essenzialmente costituito da 2 "mondi" interconnessi: una parte reale, formata da elementi materiali, e una parte virtuale dove invece sono presenti elementi inesistenti nel mondo fisico. (Boschert e Rosen 2016)
In *fig.12* viene riportato un esempio della struttura tipica di un Digital Twin di un processo.

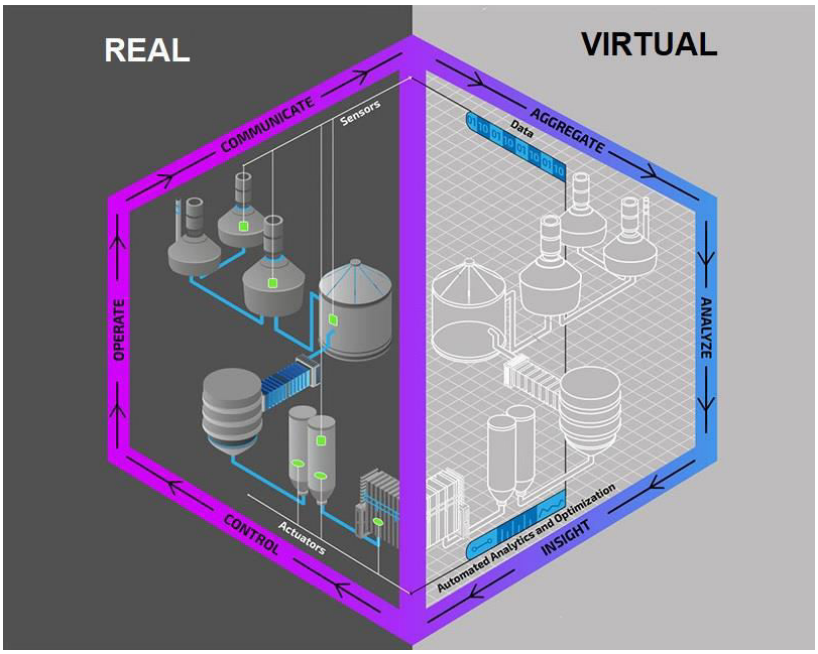


fig.12

Come si evince dalla *fig.12* il sistema Digital Twin è principalmente costituito dai seguenti elementi :

Parte REALE:

- Sensori → rilevano le varie condizioni operative della linea produttiva. Il segnale generato da questi ultimi sarà poi trasformato dal Digital Twin in dati operazionali che guideranno i "meccanismi" di previsione/ottimizzazione;

- Attuatori → utilizzati per effettuare eventuali azioni di ottimizzazione o per salvaguardare la sicurezza degli operatori quando si riscontrano situazioni anomale e pericolose.

Parte VIRTUALE:

- Dati → corrispondono ai segnali elaborati dal Digital Twin, provenienti dai sensori, oppure possono essere anche reperiti da fonti esterne. Risultano essenziali per il corretto funzionamento dell'intero sistema;

- Tecniche Analitiche → strumenti che consentono di analizzare i dati e permettono di fare poi simulazioni al fine di prevedere potenziali situazioni anomale o potenziali miglioramenti del sistema.

2.4 Funzionalità del Digital Twin

Il funzionamento di un Digital Twin si basa su uno schema ben preciso, dove si individuano 3 fasi principali:

- Fase SEE ;
- Fase THINK;
- Fase DO.

In queste 3 macro-fasi sono presenti poi sottofasi, che nell'insieme garantiscono così una corretta funzionalità del Digital Twin. (Colin 2016)

Di seguito vengono spiegate e schematizzate tali fasi (*fig.13*).

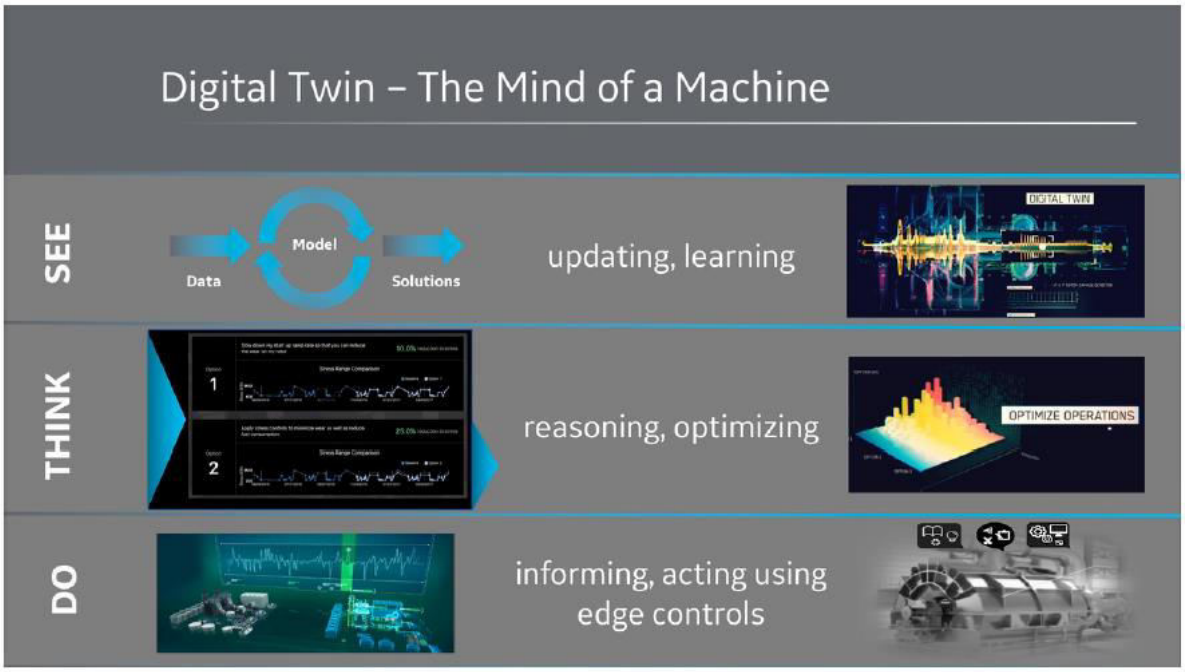


fig.13

2.4.1 Fase SEE

In questa fase il Digital Twin, osserva il processo/prodotto a cui è applicato e coglie da esso informazioni provenienti dai vari sensori presenti, al fine di rimanere sempre aggiornato e di

conseguenza replicare al meglio le condizioni dell'oggetto fisico. Le sottofasi di questa prima macro-fase, sono le seguenti:

- Creazione → il processo/prodotto "evolvendosi" nel suo funzionamento crea dei dati che vengono poi percepiti dai vari sensori;
- Comunicazione → sottofase successiva alla creazione, i sensori una volta percepiti i dati li comunicano ad un server di raccolta dati.

2.4.2 Fase THINK

In questa fase il Digital Twin, sfruttando le tecniche analitiche e gli algoritmi di apprendimento, con cui viene sviluppato (come le Reti Neurali Artificiali), simula ed analizza tutte le potenziali situazioni di malfunzionamento e anomalie che potrebbero verificarsi nel processo/prodotto. I risultati di queste analisi vengono poi utilizzati per sviluppare eventuali soluzioni. Le sottofasi di questa seconda macro-fase, sono le seguenti:

- Aggregazione → i dati raccolti nel server precedente, vengono aggregati ed elaborati;
- Analisi → i dati elaborati vengono analizzati e utilizzati poi per effettuare simulazioni;
- Previsione → in questa sottofase vengono svolte le simulazioni precedentemente descritte al fine di ricavare soluzioni a eventuali problemi riscontrati.

2.4.3 Fase DO

In questa fase il Digital Twin interagisce direttamente (in maniera automatica) o indirettamente (in maniera manuale) sul processo/prodotto reale, basandosi sulla soluzione proposta dalla fase precedente. Nel primo caso il Digital Twin effettua operazioni di miglioramento, comandando gli attuatori presenti nel sistema. Nel secondo caso invece, il Digital Twin fornisce solo informazioni su come effettuare le migliorie. Queste soluzioni proposte dovranno essere poi realizzate ed applicate al sistema reale al fine di risolvere i problemi analizzati.

2.5 Vantaggi Competitivi del Digital Twin

L'utilizzo dei sistemi di Digital Twin nel mondo industriale, garantisce alle aziende importanti vantaggi competitivi, tra cui:

- **riduzione dei difetti del prodotto e dei costi di produzione** → ottimizzazione del prodotto;
- **processi di convalida virtuale** → la fase di testing del prodotto viene fatta principalmente in modo virtuale, mantenendo comunque la connessione virtuale-fisica che consente di analizzare il rendimento di un prodotto reale in una serie di condizioni e apportare le modifiche necessarie nel mondo virtuale, per garantire che il prodotto fisico funzioni esattamente come previsto;
- **affidamento non solo sul modello fisico** → le informazioni riguardanti le prestazioni dei modelli fisici possono essere incluse nei Digital Twin, che consentono quindi di mantenere una versione fedele del modello virtuale;
- **verifica del processo di produzione** → effettuata prima di avviare la produzione, consente di eliminare costosi tempi di inattività delle macchine e dei robot. Consente inoltre di individuare quando sarà necessaria una manutenzione preventiva;
- **riduzione del time-to-market** → consente di arrivare sul mercato più velocemente della concorrenza;

Le aziende che realizzano e sfruttano i Digital Twin inoltre, si creano inoltre uno spazio di business basato sull'offerta di servizi. I Digital Twin sono infatti in grado di fornire un monitoraggio dell'operatività del prodotto/processo e permettono di continuare il testing in situazioni di utilizzo reale.

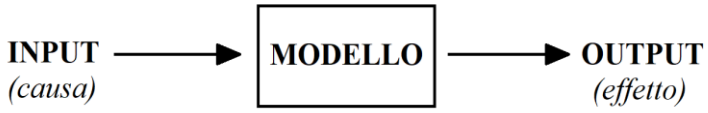
2.6 Concetto di Identificazione Parametrica

L'identificazione parametrica può essere descritta come l'arte di ricavare un modello (o i parametri che definiscono un modello) a partire da dati sperimentali.

Nella maggior parte dei casi, ogni problema di identificazione si traduce in un problema nel quale, conoscendo le varie cause i gli effetti corrispondenti, si cerca di ricostruire il modello del fenomeno fisico che li mette in relazione. Tale tipo di problema viene anche definito problema

inverso. L'esistenza di un problema inverso implica quindi l'esistenza di un problema diretto, al quale il problema inverso risulta collegato. Due problemi quindi si definiscono uno l'inverso dell'altro, quando la formulazione di uno coinvolge inevitabilmente l'altro. In genere il problema inverso risulta molto più complesso di quello diretto; ad esempio, esso può non ammette un'unica soluzione. (Mazzotti 2008)

Quando si analizza un problema, se si dispongono di sufficienti informazioni in merito, è possibile ottenere una procedura ben definita, che permette di convergere ad un'unica soluzione del problema stesso. Un problema diretto si può rappresentare nel seguente modo:



$$x \longrightarrow K \longrightarrow y$$

cioè

$$Kx = y$$

In questo caso il problema diretto consiste nell'applicare la *causa x* al *modello K* e calcolare l'*effetto y*. L'equazione precedente può però essere analizzata in altri due diversi modi; il problema diretto infatti fornisce i due seguenti problemi inversi:

1) $y/_K = x$ → dato il *modello K* e l'*effetto y*, risalire alla *causa x* ;

2) $y/_x = K$ → data la *causa x* e l'*effetto y*, realizzare un *modello K* (caso che verrà studiato in questo progetto di tesi).

2.7 White, Grey e Black - Box

Qualsiasi metodo di identificazione parametrica consiste generalmente nell'applicazione di diversi input al sistema preso in esame e nella registrazione delle uscite corrispondenti. Si avranno dunque le sequenze:

$$x(t) = x(1), x(2), \ldots, x(n) \rightarrow ingressi$$

$$y(t) = y(1), y(2), \ldots, y(n) \rightarrow uscite$$

dove *n* corrisponde al numero di misurazioni effettuate. (Mazzotti 2008)

Generalmente si possono individuare tre diverse tipologie di procedure di identificazione: (Gelfusa 2013)

- Identificazione White-Box (a scatola bianca/trasparente) → il modello viene ricavato dall'analisi delle parti che costituiscono il sistema preso in esame, attraverso le leggi fisiche che ne descrivono il comportamento. In altri termini, il fenomeno da identificare viene decomposto in parti elementari delle quali si conosce il modello matematico; così facendo si riesce poi a risalire al modello dell'intero sistema;

- Identificazione Grey-Box (a scatola grigia) → si conosce inizialmente solo parte della struttura del modello che descrive il problema fisico, poiché vi sono dei parametri incogniti che devono essere stimati. Risulta quindi un metodo intermedio tra White e Black-Box;

- Identificazione Black-Box (a scatola nera) → non si conoscono le equazioni del modello fisico e risulta impossibile stimare i loro parametri singolarmente. Si effettuano quindi delle misure sperimentali delle variabili di ingresso ed uscita, che saranno poi utilizzate esclusivamente per determinare il modello matematico. I parametri di questi modelli non assumono quindi un'interpretazione fisica, ma sono solamente un mezzo per descrivere i vari legami (relazioni) ingresso-uscita del sistema.

2.8 Considerazioni finali

In questo progetto di tesi per realizzare il Digital Twin dell'eiettore proposto, verranno analizzati algoritmi basati sull'intelligenza artificiale (Reti Neurali Artificiali e Swarm Intelligence) che riproducono al meglio le funzionalità (SEE-THINK-DO) di un Digital Twin. La selezione della tipologia di algoritmo ottimale, per il caso preso in esame, verrà effettuata nel Capitolo 3. L'identificazione parametrica del dispositivo fisico dovrà garantire un corretto modello virtuale dello stesso. L'eiettore, è uno di quei componenti dell'impianto a cui non è possibile riferire equazioni matematiche precise e chiare; sarà quindi necessario effettuare un'identificazione Black-Box.

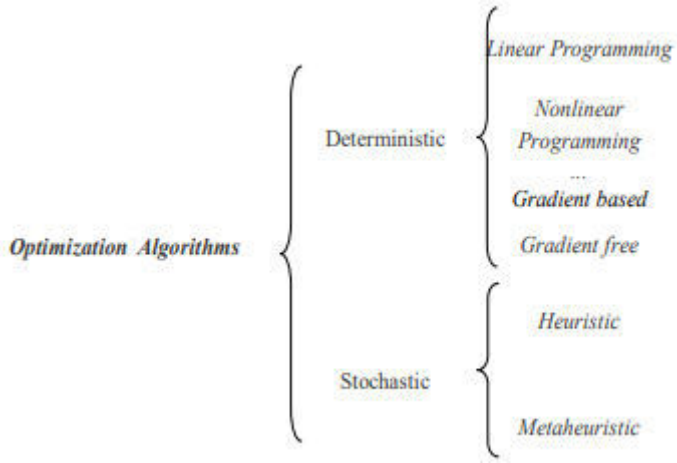
Capitolo 3

Algoritmi di Swarm Intelligence e Reti Neurali Artificiali

In questo capitolo verranno introdotte e spiegate due tipologie di algoritmi di apprendimento automatico: gli algoritmi di Swarm Intelligence e le Reti Neurali Artificiali. Verranno riportati anche i rispettivi campi di applicazione. Successivamente sarà effettuato un confronto tra le due diverse tipologie, analizzando i vantaggi e gli svantaggi di ciascuna, al fine di individuare la migliore per il caso preso in esame. Questa verrà poi utilizzata per eseguire l'identificazione parametrica dell'eiettore. Da tale studio emergerà che gli algoritmi basati sulla Swarm Intelligence, in questo caso, risultano essere migliori per diversi motivi. Infine verranno quindi analizzati 8 principali algoritmi di SI: PSO, ABC, ACO, CSA, FFA, GWO, BA e AFSA.

3.1 Tipi di algoritmi

La maggior parte degli algoritmi di ottimizzazione convenzionali o classici sono di tipo deterministico, come ad esempio il famoso metodo del simplesso nella programmazione lineare. Tali algoritmi risolvono uno specifico problema di ottimizzazione determinando la soluzione ottima tra tutte quelle ammissibili. Sebbene possano essere utilizzati per ottenere soluzioni molto accurate e puntuali, il loro utilizzo non supera l'ambito definito dal problema specifico che risolvono. Per di più per formularli sono necessarie elevate competenze matematiche. Le restrizioni appena riportate, facilitano l'uso di un'altra categoria di algoritmi: gli algoritmi stocastici. Gli algoritmi stocastici generalmente si suddividono inoltre, in euristici e metaeuristici. (Rizzo 2018)



Il termine euristico significa "trovare" o "scoprire per tentativi ed errori". Generalmente soluzioni di qualità (accettabili) per un problema complesso, possono essere individuate in tempi computazionali relativamente brevi, ma non c'è garanzia che queste ultime corrispondano con le soluzioni ottimali. Quindi tali algoritmi non garantiscono la coincidenza tra la soluzione trovata e la soluzione ottima, però risultano essere maggiormente flessibili, rispetto agli algoritmi deterministici, ottenendo soluzioni sub-ottime accettabili. Ciò risulta essere soddisfacente quando non si vogliono necessariamente trovare le soluzioni migliori, ma piuttosto le soluzioni facilmente raggiungibili.

Un ulteriore sviluppo rispetto agli algoritmi euristici sono i cosiddetti algoritmi metaeuristici. Con il termine meta si intende "a livello più alto" e in genere quindi tali algoritmi hanno prestazioni più elevate. Inoltre, tutti gli algoritmi metaeuristici utilizzano determinati trade-off di randomizzazione e ricerca locale. La randomizzazione è un buon modo per muoversi nello spazio di ricerca della soluzione, passando quindi dalla ricerca locale alla ricerca su scala globale. Per questo motivo, quasi tutti gli algoritmi metaeuristici intendono essere adatti all'ottimizzazione globale. Due componenti principali di qualsiasi algoritmo metaeuristico sono: l'intensificazione e la diversificazione (o lo sfruttamento e l'esplorazione). (Blum e Roli 2003)

Diversificazione significa generare soluzioni diverse attraverso la randomizzazione, in modo da esplorare lo spazio di ricerca su scala globale, evitando quindi che le soluzioni rimangono “intrappolate” a livello locale e, allo stesso tempo, aumentano la diversità delle soluzioni. Intensificazione invece significa concentrarsi sulla ricerca in una regione locale, sfruttando le

informazioni di una buona soluzione che si trova in tale regione. La selezione delle migliori "buoni soluzioni" garantisce che le soluzioni convergano verso il valore ottimo. Una buona combinazione di questi due componenti principali di solito assicura che l'ottimizzazione globale sia realizzabile.

Gli algoritmi metaeuristici possono essere classificati in molti modi. Un modo per classificarli è quello di distinguere la fonte di ispirazione che ha permesso appunto la realizzazione degli stessi (Fister Jr. et al. 2013). La natura ad esempio, essendo una ricca fonte di ispirazione ha ispirato molti ricercatori in molti modi. Al giorno d'oggi, la maggior parte dei nuovi algoritmi sono classificati infatti come nature-inspired. La maggior parte degli algoritmi nature-inspired si basano su alcune "caratteristiche di successo" del sistema biologico, pertanto tali algoritmi sono definiti bio-inspired. Ovviamente, non tutti gli algoritmi sono basati su sistemi biologici, alcuni algoritmi nature-ispired sono stati sviluppati ispirandosi a sistemi fisici e chimici (Physics and Chemistry Based).

$$\frac{\text{Physics algorithms}}{\text{Chemistry algorithms}} \begin{cases} \notin \text{bio-inspired algorithms} \\ \in \text{nature-inspired algorithms} \end{cases}$$

3.2 Algoritmi di Swarm Intelligence (SI) – Aspetti Generali

Tra gli algoritmi bio-ispired, una classe speciale di algoritmi è stata sviluppata traendo ispirazione dall'intelligenza dello sciame. Gli algoritmi presenti in questa classe sono quelli basati sulla Swarm Intelligence, anche detti SI-based (Fister Jr. et al. 2013):

$$\text{SI-based} \subset \text{bio-inspired} \subset \text{nature-inspired}$$

In particolare le tecniche di SI hanno lo scopo di determinare la soluzione ottima di un determinato problema, sfruttando il comportamento globale di uno "sciame" di agenti omogenei e non sofisticati. Mentre ogni agente può essere considerato come "non intelligente", l'intero sistema di più agenti mostra un comportamento di auto-organizzazione, ovvero un sistema che nel corso del tempo punta a perfezionare le sue capacità, migliorando l'organizzazione dei suoi elementi e garantendo così una sorta di intelligenza collettiva. Ogni

agente infatti condivide informazioni ed esperienze l'uno con l'altro, riuscendo, in questo modo, a risolvere compiti anche molto complessi.

La famiglia degli Swarm Intelligence è molto vasta e contiene diversi tipi di algoritmi ispirati dai comportamenti collettivi caratteristici di alcuni gruppi di insetti, come le formiche, le api e le lucciole, o di altri animali come gli stormi di uccelli, i banchi di pesci e i branchi di lupi. Negli ultimi anni si è assistito ad un'intensificazione della ricerca in tale ambito. I 3 principali algoritmi appartenenti a questa classe sono: Ant Colony Optimization (ACO), Artificial Bee Colony algorithm (ABC) e Particle Swarm Optimization (PSO) (Zedadra et al. 2018).

Nella *fig.14* sono classificati i vari algoritmi di SI, in base alla loro fonte di ispirazione.

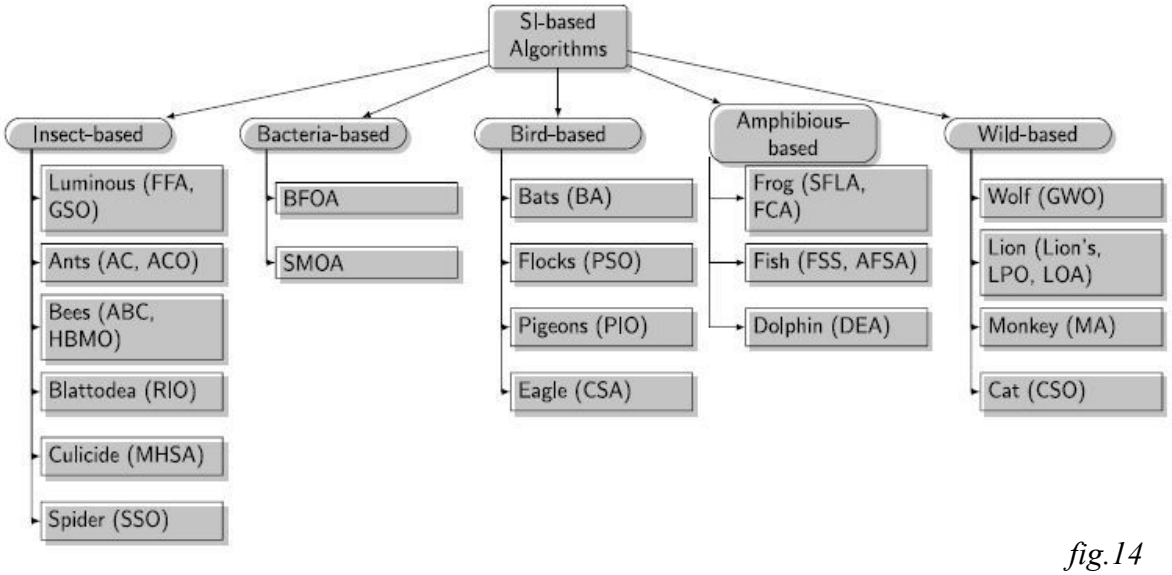


fig.14

3.2.1 Campi di applicazione SI

Gli algoritmi basati sulla SI sono divenuti tra i più popolari e utilizzati soprattutto negli ultimi anni. Ci sono molte ragioni per tale popolarità, uno dei motivi è che tali algoritmi condividono le informazioni tra più agenti, in modo che l'auto-organizzazione, la co-evoluzione e l'apprendimento durante le iterazioni possano contribuire a fornire un alta efficienza nella ricerca della soluzione ottima. Proprio per questo motivo, il loro utilizzo è cresciuto sempre di più, nel corso del tempo. Gli algoritmi di Swarm Intelligence infatti, sono utilizzati in una varia gamma di aree scientifiche :

- Engineering Design problems;
- Networking;
- Electronics application;
- Scheduling;
- Image Processing;
- Traveling Salesman Problem (TSP);
- Vehicle Routing;
- Control of Systems;
- Signal Processing;
- Data Mining;
- Economic problems;
- Management problems;
- Medical problems;
- Environmental Modelling;
- Machine Learning;
- ecc.

Successivamente verranno riportati alcuni esempi di utilizzo, nei settori sopra elencati, di vari algoritmi di Swarm Intelligence.

3.3 Reti Neurali Artificiali (ANN) – Aspetti Generali

Un sistema simile agli algoritmi di Swarm intelligence è la Rete Neurale Artificiale. Anche in questo sistema infatti sono presenti più agenti definiti "neuroni" che essendo interconnessi comunicano e scambiano informazioni tra di loro. Anche in questo caso si può parlare quindi di "intelligenza collettiva" o meglio ancora "intelligenza del sistema globale".

Le reti neurali artificiali (ANN) sono state sviluppate con lo scopo di riprodurre le attività tipiche del cervello umano. Questo ultimo è formato da un'enorme ed intrecciata rete di connessioni tra cellule, in grado di comunicare ed influenzarsi fra loro. Grazie a questa proprietà le cellule quindi, lavorando insieme, cercano di realizzare un obiettivo/compito specifico. Queste cellule nervose corrispondono ai neuroni e presentano dei prolungamenti chiamati dentriti, con molte ramificazioni. Tramite tali dentriti i neuroni sono in grado di riceve i segnali

elettrici emessi da altri neuroni. Ogni neurone è costituito anche da un assone, ossia un prolungamento filamentoso che si ramifica all'estremità dei terminali. Nei terminali si propagano i segnali elettrici destinati ai dentriti di altre cellule. La connessione presente tra un terminale ed un altro dentrite viene definita sinapsi. (Bishop 1994)

In *fig.15* è mostrato uno schema illustrativo del funzionamento di un neurone.

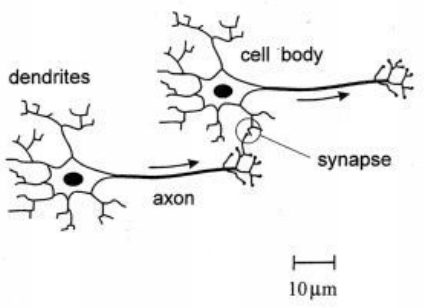


fig.15

Il cervello, pur essendo costituito da "elementi di elaborazione" molto semplici come i neuroni, risulta essere un calcolatore molto complesso e potente. Esso infatti ha la capacità di imparare e apprendere nuovi concetti, riuscendo quindi a mutare le varie connessioni fra i neuroni in base ad ogni esperienza assimilata.

Le ANN per riprodurre le capacità di elaborazione del cervello umano, devono essere realizzate secondo uno schema simile a quello precedentemente esposto. Devono quindi predisporre una rete di elementi computazionalmente semplici e devono essere in grado di apprendere e fornire determinati valori di uscite anche per ingressi non noti. Al fine di sviluppare e realizzare una ANN efficiente bisogna infine, come per un cervello umano, farla apprendere. (Marzano 2017)

Ciascun neurone artificiale di una ANN (*fig.16*) è costituito in genere da molti ingressi (n ingressi → *x1, ⋯ , xn*) e un'unica uscita (y). Ad ogni singolo ingresso viene associato un peso (*wi* ; *i* = 1, ⋯ ,n) costituito da un numero reale, che stabilisce il grado di conducibilità del rispettivo canale di ingresso. Se il peso è positivo il canale viene definito "eccitatorio", altrimenti se è negativo "inibitorio". L'uscita del neurone è definita invece da una "funzione di attivazione". Tale funzione dipende della somma pesata degli ingressi, chiamata anche "livello di attivazione" (a).

$$a = \sum_{i=1}^{n} w_i \, x_i$$

$$y = f(a) = f\left(\sum_{i=1}^{n} w_i \, x_i\right)$$

dove f è la funzione di attivazione del neurone, chiamata anche funzione di trasferimento. La funzione di trasferimento definisce quindi l'uscita di ogni neurone mediante il rispettivo livello di attivazione. (Bishop 1994)

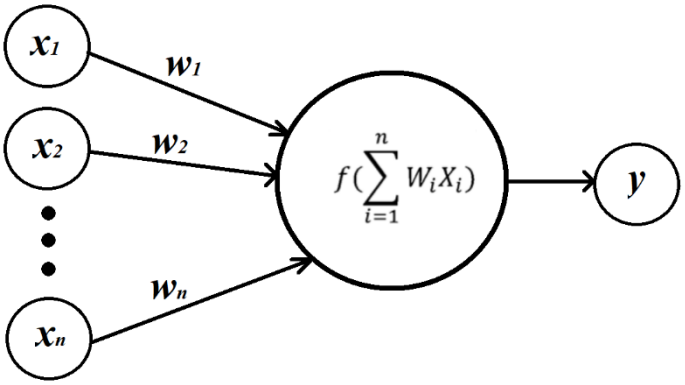


fig.16

Per effettuare l'addestramento (fase di traning) della Rete Neurale Artificiale è necessario avere a disposizione un dataset (insieme di dati di addestramento), che verrà somministrato in input alla rete e un insieme di test per verificare la corretta riuscita dell'addestramento stesso. Durante la fase di training, dopo aver fornito in input alla ANN l'insieme dei dati di addestramento, quest'ultima genera come output un vettore di risultati. Attraverso una funzione definita funzione di costo, o funzione errore, si calcola appunto l'errore (inteso come differenza) tra i risultati di output e i risultati desiderati. Rimanendo in fase di addestramento, la rete cercherà di minimizzare questo "errore", modificando il valore dei suoi parametri interni (pesi *wi*). Questa procedura viene ripetuta fino a quando l'uscita o le uscite della rete non forniscono un errore con valore al di sotto di una soglia prestabilita. Questa modalità di addestramento è definita "supervisionata".

Dopo l'addestramento si passa ad una fase di testing della rete: si valutano le sue prestazioni con il test set. I dati appartenenti al test set non devono essere mai analizzati dalla rete prima dell'addestramento. Solitamente di tutti i dati che si hanno a disposizione è meglio allocarne un numero più elevato per il training set ed un numero più ridotto per il testing set. (Gallo 2007)

Una ANN viene definita stratificata, quando è formata da diversi strati di neuroni. Ogni neurone di ciascuno strato è connesso con quelli dello strato successivo, senza però alcuna connessione tra neuroni facenti parte del medesimo strato. Una rete viene definita feedforward quando i segnali di ingresso attraversano la rete in un'unica direzione, in particolare spostandosi sempre "in avanti", ossia verso l'uscita. In questo ultimo tipo di reti solitamente possono essere presenti uno o più strati nascosti (hidden layers), ossia strati di neuroni che non si interfacciano direttamente con l'esterno. (Marzano 2017)

In *fig.17* viene data una rappresentazione grafica di una rete feedforward stratificata a 3 strati con 2 strati nascosti, 2 ingressi e 1 uscita

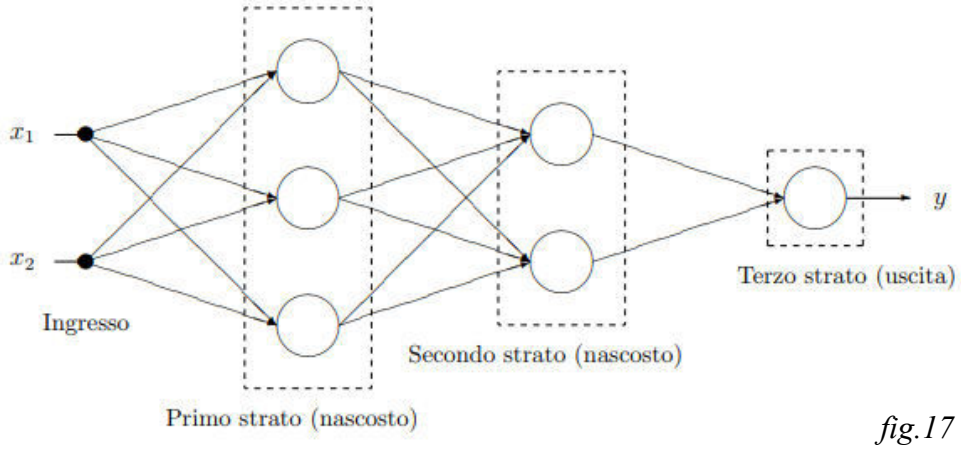


fig.17

3.3.1 Campi di applicazione ANN

Le Reti Neurali Artificiali (ANN) vengono impiegate in tutti quei casi dove l'analisi statistica di tutte le variabili di un problema risulti essere difficoltosa o onerosa in termini di calcolo, ma soprattutto quando non siano chiare a priori, quali relazioni deterministiche esistano tra le diverse variabili che caratterizzano il problema, oppure nei casi in cui non esistano modelli analitici in grado di affrontare il problema.

In genere, una ANN può essere utilizzata per la risoluzione di quattro principali categorie di problemi, ossia per:

- classificare dei dati in vari gruppi;
- riconoscere modelli e schemi all'interno di grandi quantità di dati;
- effettuare predizioni basate sui dati di ingresso forniti alla rete;
- ottimizzare un risultato già ottenuto con altre metodologie.

Premesso che le applicazioni delle ANN sono talmente numerose da rendere difficile ogni tentativo di classificazione, di seguito viene riportata una serie di aree di utilizzo:

- Cognitive and Emotional Robotics;
- Neurorehabilitation Robotics;
- Audio-visual voice recognition;
- Quality Control of industrial production;
- Cybersecurity;
- Advertising;
- Clustering
- Image Processing;
- Automotive;
- Manufacturing;
- Healthcare;
- Data Science;
- Economy and Finance;
- ecc.

3.4 Confronto tra ANN e SI

In questa sezione verranno confrontate le due tipologie di algoritmi precedentemente evidenziate (ANN e SI) attraverso un'analisi accurata dei vantaggi e svantaggi (analisi V/S) relativi alle stesse. L'obiettivo sarà quello di individuare tra le due, la tipologia più adatta al caso proposto da studiare e sviluppare, ossia la più adatta per effettuare una corretta identificazione parametrica dell'eiettore.

3.4.1 Analisi V/S - SI

In primo luogo si riporta l'analisi V/S degli algoritmi di Swarm Intelligence (Kordon 2010):

VANTAGGI - Swarm Intelligence

1 - Derivative-Free Optimization (Ottimizzazione senza derivate)

La ricerca di una soluzione ottimale non si basa su funzioni derivate, ma su diversi meccanismi di interazione sociale tra individui artificiali. In questo modo le possibilità

di convergere e rimanere “ancorati” nelle soluzioni locali (non globali) sono ridotte in maniera significativa, tuttavia non sono eliminate.

2 - Robustezza

Gli algoritmi basati sulla popolazione sono più robusti al fallimento individuale. Le scarse prestazioni di diversi membri dello sciame non è un pericolo per il rendimento complessivo. Il comportamento collettivo compensa infatti i singoli deficit e la soluzione ottimale si trova indipendentemente dalle variazioni delle prestazioni individuali.

3 – Dinamicità

Probabilmente il più grande vantaggio dell'intelligenza dello sciame è la sua capacità di operare in un ambiente dinamico. Lo sciame può seguire continuamente il percorso anche per un’ottimizzazione in rapida evoluzione. In linea di principio, non vi è alcuna differenza significativa nel funzionamento dell'algoritmo in stato stazionario o in modalità dinamica.

4 - Facilità di marketing e di implementazione

I principi dello sciame, ispirato alla biologia, sono facili da comunicare a un vasto pubblico di potenziali utenti e non c'è bisogno di un grande background matematico o statistico. Inoltre, l’implementazione sia dell'ACO che, in particolare, del PSO (due particolari algoritmi di SI) su qualsiasi ambiente software risulta essere semplice. I parametri di ottimizzazione solitamente sono pochi e facili da capire e da regolare.

5 - Basso costo totale

Il basso costo di implementazione e di marketing, uniti poi ai costi di manutenzione che risultano essere potenzialmente bassi data l'adattabilità degli algoritmi alle mutevoli situazioni operative, garantiscono infine un basso costo totale.

6 - Parallelizzazione

Più agenti del sistema “Swarm” possono essere messi in parallelo facilmente in modo che l'ottimizzazione su larga scala diventi più pratica e rapida dal punto di vista dell’

implementazione. Questa proprietà definisce quindi la capacità degli agenti, componenti lo sciame, di effettuare una moltitudine di azioni in luoghi differenti e nello stesso momento. Ciò è fondamentale, poiché consente di sviluppare sistemi più flessibili capaci quindi di auto-organizzarsi in gruppi, che considerano simultaneamente diversi aspetti di un particolare problema.

7 - Scalabilità

Quando si verifica un incremento della dimensione del problema (o della quantità delle variabili), l'algoritmo riesce a conservare la propria funzione senza dover riesaminare le modalità con cui gli agenti interagiscono. Grazie al principio di scalabilità un sistema può incrementare la sua performance semplicemente aumentando la sua dimensione e, soprattutto, senza la necessità di alcuna riprogrammazione.

8 - Tolleranza d'errore

Proprietà che deriva dalla natura decentralizzata ed auto-organizzata dei sistemi di SI. Data la presenza di molti individui intercambiabili, ogni individuo debole può essere facilmente eliminato e sostituito da un altro pienamente funzionante. Tale principio quindi consente di sorvolare eventuali errori, senza però creare problemi particolari nella computazione.

SVANTAGGI - Swarm Intelligence

1 - Convergenza prematura

Alcuni tipi di SI (come il PSO) di solito soffrono di convergenza prematura quando si verificano problemi di ottimizzazione multipla. Alla base di questo problema c'è il fatto che, per il PSO ad esempio, le particelle convergono in un unico punto, che si trova sulla linea tra il le migliori posizioni a livello globale e le migliori posizioni personali degli agenti artificiali.

2 - Sensibilità ai parametri

Un altro svantaggio degli algoritmi di SI è che le prestazioni generalmente sono molto sensibili all'impostazione dei vari parametri. Ad esempio, aumentando il valore del peso d'inerzia, *w* (nel PSO), aumenterà la velocità delle particelle con conseguente maggiore

esplorazione (ricerca globale) e minore concentrazione (ricerca locale). La regolazione quindi dei parametri non è un compito facile e varia da problema a problema.

3 - Limitate applicazioni industriali

Il numero di applicazioni industriali basate sull'intelligenza degli sciami è ancora limitato. Tuttavia la situazione può cambiare molto rapidamente con un marketing di successo e l'identificazione di classi di applicazione adeguate.

4 - Nuovo settore di ricerca

Rispetto alla maggior parte degli altri approcci all'intelligenza computazionale, gli Swarm Intelligence hanno una breve storia di ricerca e ancora varie problematiche teoriche.

5 - Non Immediatezza

Sistemi di sciami complessi con ricche gerarchie richiedono tempo. Più complesso è lo sciame, più tempo ci vuole per raggiungere una soluzione ottimale o sub-ottimale.

3.4.2 Analisi V/S - ANN

Successivamente viene riportata l'analisi V/S delle Reti Neurali Artificiali (Tu 1996),(Riccò 2013):

VANTAGGI - Reti Neurali Artificiali

1 – Parallelizzazione

Le ANN lavorano in parallelo e riescono ad elaborare una grande quantità di dati, a differenza dei calcolatori tradizionali dove ogni dato viene processato in serie. Ciò implica che, nel caso in cui alcune unità di elaborazione del sistema (ANN) risultassero danneggiate, questo ultimo subirebbe dei peggioramenti in termini di prestazioni, ma raramente si arresterà in modo anomalo.

2 – Robustezza

Le ANN hanno la capacità di resistere al rumore, cioè, anche nel caso in cui alcune connessioni della rete vengano danneggiate, oppure quando siano presenti disturbi nel segnale di ingresso, o nei vari canali di trasmissione, esse continuano a funzionare correttamente. In qualsiasi caso, le connessioni danneggiate potranno poi essere riaddestrate e tornare quindi ad essere funzionali.

3 – Flessibilità

Una rete neurale può essere adoperata in molti ambiti e per molti scopi diversi. Quest'ultima infatti è in grado di apprendere nuovi compiti, in base all'esperienza interiorizzata. Ciò non implica il fatto che ogni rete si possa impiegare per risolvere tutti i tipi di compiti, ma evidenzia il fatto che l'utilizzatore, di tale rete, non debba per forza conoscere le soluzioni che contraddistinguono il problema analizzato.

4 – Capacità di adattamento

Le ANN sono in grado di adattarsi alle mutevoli condizioni dell'ambiente in cui lavorano. Per farlo, regolano in maniera autonoma i propri pesi sinaptici. Tali regolazioni possono avvenire anche in tempo reale, in particolare nei casi in cui le reti lavorano in condizioni non stazionarie. In linea di massima, più l'ANN è adattiva, più le sue prestazioni risulteranno migliori in ambienti dinamici.

5 – Sviluppo possibile tramite diversi algoritmi di addestramento

Il principale algoritmo di addestramento è il "back-propagation", anche se esistono e sono stati sviluppati una molteplicità di algoritmi alternativi per la formazione delle reti. L'algoritmo di formazione ottimale, per un particolare problema, può essere solo determinato empiricamente. Le reti tendono a funzionare al meglio quando i dati da usare per l'addestramento sono stati normalizzati.

SVANTAGGI - Reti Neurali Artificiali

1- "Black-Box" - capacità limitata di identificare possibili relazioni causali tra variabili

Il difetto maggiore è sicuramente il comportamento "Black-Box", ciò denota la difficoltà di poter analizzare la computazione delle ANN in modo chiaro: le reti sono in grado di fornire output corretti, o sufficientemente corretti, ma non consentono di analizzare in maniera dettagliata, i singoli stadi di elaborazione che determinano tali risultati.

Lo sviluppatore del modello della rete imposta i dati di addestramento e poi essenzialmente lascia che la stessa si "alleni" autonomamente e determini quali siano le variabili di input più importanti.

Non è facile determinare quali siano le variabili con contributi più importanti per un particolare output, inoltre una ANN può contenere una serie di variabili non importanti che lo sviluppatore non riesce ad apprezzare.

Non esiste nessun criterio ben definito per interpretare i pesi di collegamento tra i vari neuroni dei diversi strati, sono però state sviluppate procedure per aumentare la loro comprensione. Una di queste tecniche è quella di addestrare la rete rimuovendo di volta in volta un solo nodo di una specifica variabile di ingresso e poi osservare l'effetto sulle prestazioni della rete.

2 – Difficoltà applicative in vari settori

Le ANN possono risultare più difficili da utilizzare sul campo rispetto al altri modelli sviluppati usando gli stessi dataset. Inoltre l'output di una ANN può essere più difficile da interpretare rispetto quello di un altro modello.

3 – Risorse di calcolo maggiori per la Modellazione

Lo sviluppo di modelli di reti neurali è un'attività computazionale intensiva che richiede un tempo di calcolo elevato. Con hardware standard per personal computer e l'algoritmo di back-propagation, spesso ci possono volere da ore a giorni a settimane prima che una rete converga verso uno stato di apprendimento ottimale con il minimo errore.

Varianti più rapide della back-propagation sono state sviluppate, ma potrebbero ancora richiedere grandi quantità di tempo di addestramento.

4 – Lunghi tempi di addestramento

Le reti neurali come detto precedentemente, necessitano di una fase di addestramento necessaria anche ad ottimizzare i pesi delle connessioni tra i diversi strati. Se il numero dei dati di training e delle variabili analizzate è elevato, quest'ultima fase potrebbe richiedere molto tempo. Questo inoltre, per la maggior parte delle volte, risulta difficilmente prevedibile.

5 - Sovradattamento (Overfitting)

La capacità di un'ANN di modellare le interazioni e le non linearità, implicitamente può anche essere uno svantaggio perché può portare ad un sovraadattamento.

Il fenomeno dell' Overfitting accade nei casi in cui l'apprendimento è stato effettuato in maniera eccessiva o nel caso in cui viene fornito uno scarso numero di esempi di allenamento. In questi casi infatti il modello potrebbe adattarsi a caratteristiche che sono specifiche solamente del training set, ma che non trovano riscontro nel resto dei casi. Le prestazioni quindi sui dati di allenamento, come la capacità di adattarsi/prevedere, aumenteranno, mentre le prestazioni sui dati non visionati risulteranno essere peggiori.

In generale, l'overfitting può essere evitato in tre modi principali:

- limitando il numero di nodi nascosti;
- aggiungendo un termine di penalizzazione alla funzione obiettivo per i grandi pesi;
- limitando la quantità di dati di addestramento utilizzando la Convalida Incrociata (Cross Validation).

Sebbene tutti e tre i metodi sono potenzialmente utili, il più popolare è l'ultimo riportato, poiché è computazionalmente meno intensivo.

6 - Sviluppo empirico dei modelli e problemi metodologici

La modellazione tramite ANN è relativamente una nuova tecnica emersa da una varietà di discipline, e un numero di importanti questioni metodologiche restano da risolvere. È difficile concludere che si sia sviluppata la migliore rete neurale artificiale possibile per una particolare applicazione data la vastità di algoritmi di addestramento disponibili. Esistono poche linee guida per predeterminare quale struttura di rete e quali algoritmi di addestramento sono maggiormente adatti per un determinato problema, quindi la riuscita di una rete dipende molto dall'esperienza dello sviluppatore.

3.4.3 Considerazioni finali

Di seguito viene riportata una tabella riassuntiva che evidenzia il confronto tra le caratteristiche delle due tipologie di algoritmi precedentemente analizzate (*tab.2*).

Caratteristiche	Artificial Neural Network	Swarm Intelligence
Parallelizzazione	SI	SI
Robustezza	SI	SI
Capacità di Adattamento	SI	SI
Comportamento "Black-Box"	SI	NO
Carico Computazionale elevato	SI	NO
Facilità di implementazione	NO	SI
Costi di implementazione	Medio-Alti	Bassi
Applicazioni Industriali	Elevate	Limitate
Difficoltà Applicative	Riscontrate in vari settori	Non riscontrate

tab.2

Sebbene le Reti Neurali Artificiali, analizzando la letteratura, rispetto agli Swarm Intelligence, siano state utilizzate maggiormente, per effettuare identificazioni parametriche e analisi dei sistemi dove sono presenti eiettori:

(Nasseh et al. 2007) hanno sviluppato un modello di ANN per la previsione della caduta di pressione negli scrubber Venturi. Gli autori hanno progettato tre ANNs indipendenti. Dopodiché hanno confrontato i risultati di queste, con i risultati calcolati da altri modelli. Tale analisi ha evidenziato che i risultati delle ANNs risultavano essere maggiormente simili ai dati sperimentali; (Sözen e Arcaklioğlu 2007) hanno utilizzato le reti neurali artificiali per effettuare un'analisi termodinamica del ciclo di espulsione-assorbimento di sistemi termici. In particolare gli autori, tramite questo nuovo metodo, sono riusciti a calcolare le perdite di energia nel sistema, evitando di effettuare l'analisi termodinamica classica che utilizza complesse equazioni differenziali e complessi programmi di simulazione; (Haoran e Wenjian 2014) hanno sviluppato una ANN per stimare la pressione di uscita da un eiettore, dati vari stati di ingresso. Gli autori hanno quindi proposto un metodo alternativo di modellazione degli espulsori, rispetto ai tradizionali (modello termodinamico e modello CFD). Questa ricerca inoltre ha posto le basi per le strategie di controllo della costruzione di un sistema di refrigerazione basato su eiettori, utilizzando metodi di apprendimento automatico; (Wang, Cai, e Wang 2016) hanno modellato un sistema di climatizzazione ibrido ad eiettore utilizzando le

reti neurali artificiali al fine di prevedere le prestazioni di questo ultimo. In primo luogo, gli autori hanno adoperato tre diversi tipi di reti neurali (MLP – RBF – SVM). Successivamente confrontando i risultati ottenuti, la rete di tipo MLP è risultata essere la più performante, ossia forniva una differenza minore tra il valore previsto e quello misurato;

in questo progetto di tesi verranno utilizzati vari algoritmi di Swarm Intelligence.

Questa decisione è motivata soprattutto dal fatto che, come visto nell'analisi V/S – SI, gli algoritmi di Swarm Intelligence ad oggi hanno ancora limitate applicazioni industriali rispetto alle Reti Neurali Artificiali: l'obiettivo sarà quindi quello di creare un modello digitale in grado di rappresentare al meglio il funzionamento dell'eiettore e poi divulgare tali risultati al fine di accrescere le sperimentazioni in materia. Altre tre importanti motivazioni ricadono nel fatto che l'implementazione di varie tipologie di algoritmi di SI, risulta non essere complessa e il costo associato a tali tipi di progetti non è elevato. Inoltre si è tenuto maggiormente conto anche del fatto che le ANN sono sostanzialmente delle "Black-Box", ciò significa che le reti non permettono di esaminare le relazioni che si instaurano tra le varie variabili del problema, nei singoli stadi di elaborazione, ma forniscono solo la soluzione ottenuta. Queste ultime per giunta a differenza degli algoritmi di SI, presentano una complessità computazionale elevata, ossia necessitano di risorse di calcolo maggiori.

3.5 Principali Algoritmi di Swarm Intelligence

In questa sezione verranno indicati e descritti i principali algoritmi basati sulla Swarm Intelligence. Si analizzeranno quindi gli algoritmi maggiormente utilizzati e i vari campi applicativi saranno evidenziati di seguito. Per ogni algoritmo verranno riportati inoltre i principali parametri da inizializzare, lo pseudocodice ed i relativi vantaggi e svantaggi.

3.5.1 PSO – Particle Swarm Optimization

L'algoritmo PSO è il principale algoritmo di Swarm Intelligence. È stato sviluppato nel 1995 da J. Kennedy e R. Eberhart con il fine di simulare graficamente la coreografia degli stormi di uccelli per scoprire quindi le regole con cui essi volano in maniera sincronia, decidono di cambiare direzione in maniera improvvisa e si disperdono per poi riunirsi. Solo in seguito è

stato adoperato come metodo per risolvere vari problemi di ottimizzazione. (Ab Wahab, Nefti-Meziani, e Atyabi 2015)

Ciascun volatile (particella) appartenente ad uno stormo (sciame artificiale) deve perseguire nel modo migliore determinati obiettivi, come ad esempio ricercare il cibo o fuggire dai predatori. Per raggiungere tali propositi, questo ultimo sfrutta le informazioni sulla propria posizione e velocità, nonché sulla direzione e velocità prese dall'intero stormo. Ciascun volatile quindi stabilisce il proprio comportamento attenendosi anche alle scelte degli altri. (Zuccon 2017)

Il comportamento di ogni particella, in questo caso rappresentata da un volatile, nel momento in cui individua una fonte di cibo, cioè un obiettivo da raggiungere, è simulato secondo tre regole, come mostrato in *fig.18*. Il primo comportamento è la separazione dove ogni agente cerca di allontanarsi dal gruppo per raggiungere l'obiettivo. Il secondo è l'allineamento in cui ogni agente si sposta, seguendo la direzione media dei suoi vicini. Il terzo è la coesione dove ogni agente cerca di andare verso la posizione media dei suoi vicini.

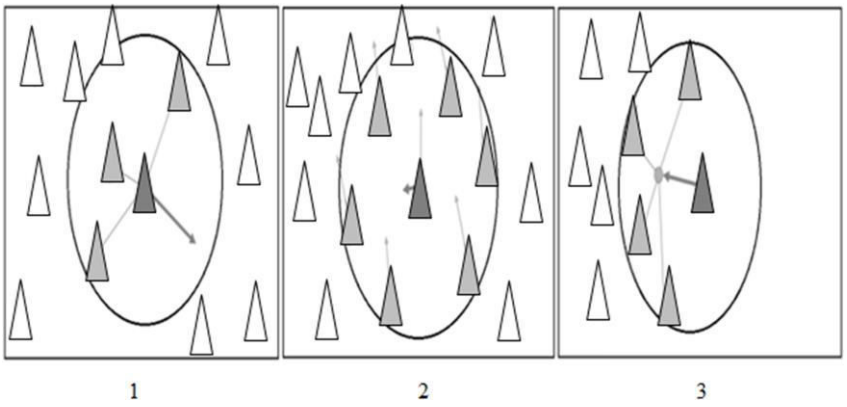


fig.18

Si possono quindi distinguere due strategie di ricerca: una basata sull'esplorazione, quindi legata alla separazione della singola particella dal gruppo, ed un'altra basata sullo sfruttamento, legata alla coesione e quindi all'utilizzo dei "successi" delle altre particelle del gruppo.

Inizialmente un numero predeterminato di particelle viene collocato all'interno della regione ammissibile delle soluzioni, in maniera arbitraria. Tale regione corrisponde allo spazio di ricerca della soluzione ottima.

Ciascuna particella ha tre parametri principali:

- Posizione
- Velocità → determina il tasso di variazione della posizione;
- Fitness → valore della funzione obiettivo nella posizione attuale della particella. Il valore di fitness è una misura di quanto sia buona la soluzione, questo perché ogni particella dello sciame è una potenziale soluzione al problema di ottimizzazione preso in esame e ciascuna di esse esplora il dominio di ricerca spostandosi.

Ogni particella si sposta analizzando la propria esperienza e l'esperienza collettiva dello sciame. Mentre si muovono nello spazio di ricerca, le particelle memorizzano la posizione della migliore soluzione che hanno trovato. Ad ogni iterazione dell'algoritmo, ogni particella si muove con una velocità che è una somma ponderata di tre componenti:

- la vecchia velocità;
- una componente di velocità che guida la particella verso la posizione dove in precedenza ha trovato la soluzione migliore;
- una componente di velocità che guida la particella verso la posizione dove le particelle vicine hanno trovato la soluzione migliore fino a quell'istante.

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni → quante volte far reiterare l'algoritmo;
- nr. di particelle;
- coeff. C1 → parametro che determina l'intensità della forza che attrae la singola particella verso la migliore posizione da essa trovata;
- coeff. C2 → parametro che determina l'intensità della forza che attrae la singola particella verso la migliore posizione trovata dalle particelle vicine;
- peso di inerzia "w" (eventuale) → regola l'influenza della velocità precedente di ogni singola particella, nel calcolo della nuova velocità;
- fattore di costrizione "K" (eventuale) → aumenta la possibilità di convergenza, limitando la nuova velocità di esplorazione di ciascuna particella.

L'algoritmo PSO ha molti vantaggi. È semplice da implementare, ha un numero limitato di parametri da inizializzare, ed è facilmente parallelizzabile per l'elaborazione simultanea. Garantisce inoltre un tempo di calcolo relativamente breve. Come svantaggi però presenta una prematura convergenza in punti mediamente ottimali, oltre ad avere una lenta convergenza in fase di ricerca raffinata. (Ab Wahab, Nefti-Meziani, e Atyabi 2015)

In *fig.19* viene riportato lo pseudocodice del PSO.

Particle Swarm Optimization

1. *Objective function* $f(x)$.
2. *Initialize parameters* c_1, c_2, w_{max}, w_{min}, *and population size* $nPop$.
3. *Generate an initial population of particles.*
4. *Evaluate the fitness of each particle and set all initial positions as* $P_{Best_{x_i}}$.
5. **while** $(t < MaxGeneration)$ or (!stop criterion)
6. *Select the* G_{Best} *particle in the swarm, which has the minimum fitness value.*
7. **for** $i = 1: nPop$
8. *Calculate the velocity of particle* x_i.
9. *Update the position of particle* x_i.
10. **end for** i
11. **for** $i = 1: nPop$
12. *Evaluate the fitness of updated particle* x_i.
13. **if** $f(x_i) < f(P_{Best_{x_i}})$,
14. *Then set current position as* $P_{Best_{x_i}}$.
15. **end if**
16. **end for** i
17. *Find the best particle.*
18. **end while**

fig.19

Campi di applicazione PSO

Il PSO è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Control of Systems: (Jun e Kanyu 2011) hanno utilizzato il PSO per progettare il controllore PID al fine di controllare la temperatura in un sistema HVAC (Heating, Ventilation and Air Conditioning). Gli autori hanno sviluppato un controllore PSO-PID e ne hanno verificato le prestazioni attraverso l'applicazione di tale controllore in una camera d'albergo. I risultati dimostrano che il controllore PSO-PID raggiunge prestazioni migliori di quello classico (PID) soprattutto in un ambiente mutevole; Biomedical application: (Eberhart e Xiaohui Hu 1999) hanno sviluppato il PSO al fine di analizzare due tipologie di tremore umano: il tremore classico e il mordo di Parkinson. In questo caso il PSO è stato utilizzato in combinazione ad una ANN per distinguere i pazienti sani, da quelli con tremori riconducibili ai 2 casi precedenti. Tale metodo è risultato veloce e molto accurato; Scheduling: (Al-anzi e Allahverdi 2006) hanno

affrontato il problema della schedulazione delle operazioni di assemblaggio dei computer, con l'obiettivo di minimizzare i ritardi che si possono presentare nelle varie fasi. Gli autori hanno implementato vari algoritmi tra cui il PSO. Successivamente sono stati condotti esperimenti computazionali per confrontare le euristiche proposte. I risultati mostrano la maggior efficienza del metodo euristico PSO rispetto al Tabu Search e all' EDD (Earliest Due Date); Image Processing: (Gorai e Ghosh 2011) hanno utilizzato il PSO per effettuare un miglioramento della qualità di un immagine a colori, massimizzando il contenuto informativo della stessa. La qualità dell'immagine viene migliorata da una funzione di trasformazione parametrizzata, in cui i parametri sono ottimizzati dal PSO. L'algoritmo è stato testato su diverse immagini a colori e i risultati successivamente sono stati confrontati con altre due tecniche di miglioramento. Gli autori hanno constatato che il PSO produce risultati migliori rispetto agli altri due metodi.

3.5.2 ABC – Artificial Bee Colony

L'algoritmo ABC (Artificial Bee Colony) è una tecnica di ottimizzazione che simula il comportamento intelligente di nutrimento di una colonia di api, ed è stato sviluppato e proposto da Karaboga nel 2005.

Lo sciame di api può svolgere con successo vari compiti attraverso la cooperazione sociale. Nell'algoritmo ABC, sono presenti tre tipi di api che svolgono diversi compiti: api impiegate (Employed Bees), api osservatrici (Onlooker Bees) e api scout (Scout Bees). Le api impiegate individuano le fonti di cibo e ne condividono le relative informazioni con le api osservatrici. Le api osservanti tendono a selezionare le buone fonti di cibo tra quelle trovate dalle api impiegate. Le api scout sono alcune delle api impiegate, che abbandonano le proprie fonti di cibo trovate e ne cercano nuove. (Hamad 2018)

In maniera più dettagliata viene riportata una spiegazione dei compiti svolti da ogni tipologia di ape:

- Api impiegate

  Ad ogni ape impiegata è assegnata una fonte di cibo che corrisponde ad una possibile soluzione ottima del problema. Essa è responsabile della raccolta del nettare da tale fonte di cibo e successivamente vola indietro verso l' alveare per condividere le

informazioni di raccolta, compresa l'ubicazione, la redditività del nettare in tale fonte, ecc., con le api osservatrici;

- Api osservatrici

  Le api impiegate svolgeranno un processo chiamato "waggle dance" per condividere, con le api osservatrici, le informazioni delle proprie fonti di cibo individuate. Dopodiché, ogni ape osservatrice sceglierà una fonte di cibo in base alla redditività di nettare (parametro espresso dalla funzione “fitness”). Più la fonte di cibo è redditizia, più alta è la probabilità che le api osservatrici scelgano tale fonte;

- Api Scout

  Se una fonte di cibo non ha più un nettare redditizio, le api impiegate abbandoneranno quella fonte di cibo e diventeranno api scout. Tutte le api scout scelgono poi, in maniera casuale, una nuova fonte vicino al loro alveare.

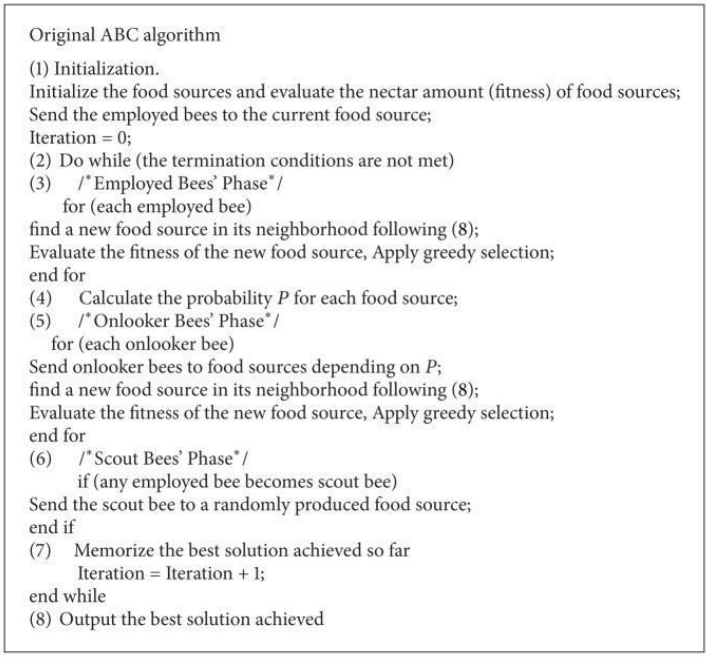
Nel complesso, le api scout effettuano l'esplorazione, le api impiegate e le api osservatrici svolgono invece il compito di sfruttamento della fonte di cibo trovata. Questo processo di foraggiamento, ossia di ricerca del nettare, viene ripetuto un certo numero di volte (iterazioni) al fine di garantire una ricerca continua di fonti di cibo sempre migliori.

I parametri da inizializzare per l’implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di api

Questo bassissimo numero di parametri da inizializzare fornisce all’ABC un grande vantaggio competitivo rispetto gli altri algoritmi di SI. L’ABC inoltre come il PSO risulta essere facilmente implementabile, inoltre può essere ibridizzato in maniera semplice con altri algoritmi e per questo è considerato un algoritmo molto flessibile. Garantisce una rapida convergenza e una buona capacità di ricerca globale. Di contro però si potrebbe incorrere in una convergenza a volte troppo prematura e per giunta presenta una scarsa capacità di ricerca locale. (Ab Wahab, Nefti-Meziani, e Atyabi 2015)

In *fig.20* viene riportato lo pseudocodice dell'ABC.



```
Original ABC algorithm
(1) Initialization.
Initialize the food sources and evaluate the nectar amount (fitness) of food sources;
Send the employed bees to the current food source;
Iteration = 0;
(2) Do while (the termination conditions are not met)
(3)    /*Employed Bees' Phase*/
       for (each employed bee)
find a new food source in its neighborhood following (8);
Evaluate the fitness of the new food source, Apply greedy selection;
end for
(4)    Calculate the probability P for each food source;
(5)    /*Onlooker Bees' Phase*/
    for (each onlooker bee)
Send onlooker bees to food sources depending on P;
find a new food source in its neighborhood following (8);
Evaluate the fitness of the new food source, Apply greedy selection;
end for
(6)    /*Scout Bees' Phase*/
       if (any employed bee becomes scout bee)
Send the scout bee to a randomly produced food source;
end if
(7)    Memorize the best solution achieved so far
       Iteration = Iteration + 1;
end while
(8) Output the best solution achieved
```

fig.20

Campi di applicazione ABC

L'algoritmo ABC è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Electronics: (Rao 2010) ha presentato un nuovo metodo basato sull'ABC per il posizionamento dei condensatori nei sistemi di distribuzione con l'obiettivo di minimizzare le perdite di potenza. Per dimostrare la validità dell'algoritmo proposto, l'autore ha effettuato simulazioni al computer su 69 linee di distribuzione e confrontato i risultati con un altro approccio disponibile in letteratura. L'ABC ha superato gli altri metodi in termini di qualità della soluzione e di efficienza computazionale; Image Processing: (Chidambaram e Lopes 2009) hanno applicato l'ABC per effettuare il riconoscimento degli oggetti nelle immagini, al fine di trovare un modello o un'immagine di riferimento (template) dell'oggetto stesso. I risultati degli esperimenti svolti con immagini in scala di grigi e a colori, mostrano che la ABC è più veloce di altri algoritmi evolutivi con precisione comparabile; Scheduling: (Pansuwan, Rukwong, e Pongcharoen 2010) hanno utilizzato l'ABC per la pianificazione MMMS (Multi-stage Multi-machine Multi-product Scheduling) della produzione e dell'assemblaggio di prodotti complessi, basata sulla filosofia "Just In Time". L'algoritmo proposto è stato progettato per ridurre al minimo il costo delle eventuali penali per i ritardi; Data Mining: (Karaboga, Okdem, e Ozturk 2010) hanno proposto un nuovo approccio di clustering per le reti di sensori wireless al fine di

mantenere al minimo l'esaurimento energetico della rete, utilizzando l'ABC. L'algoritmo proposto fornisce quindi un'organizzazione ottimale dei cluster che minimizza il consumo di energia richiesta per le comunicazioni tra i sensori in una specifica rete.

3.5.3 ACO – Ant Colony Optimization

La tecnica di ottimizzazione delle colonie di formiche (ACO) si ispira al modo in cui le colonie di formiche trovano la via più breve tra la fonte di cibo ed il loro nido. Marco Dorigo fu il primo a sviluppare e implementare questo tipo di algoritmo nel 1992.

I biologi hanno studiato a lungo e in modo approfondito come le colonie di formiche riescano collettivamente a risolvere problemi difficili in modo naturale, mentre risultano essere troppo complessi per una singola formica.

Le formiche quando si spostano rilasciano una sostanza volatile chiamata "feromone". Quando una formica deve effettuare una scelta di percorso, ad esempio in un bivio, quest'ultima può scegliere la strada in cui la concentrazione di feromone è maggiore. Dal momento che il percorso più breve verrà cosparso in maniera maggiore di feromoni rispetto quello più lungo, dopo un po' di tempo, una grande maggioranza di formiche della colonia, viaggerà su quella rotta. (Saka et al. 2013)

La *fig.21* rappresenta la fase iniziale, la fase intermedia e i risultati finali dell'algoritmo ACO. La *fig.21.1* illustra l'inizio dell'algoritmo, dove un agente artificiale (formica) inizia a muoversi e ad esplorare un possibile percorso in modo casuale, dal nido verso la sorgente, dopodiché ritorna indietro. La *fig.21.2* illustra le iterazioni intermedie di esecuzione, quando le formiche scoprono tutti i molteplici percorsi tra il nido e la fonte di cibo. Il percorso più breve viene successivamente identificato e le formiche quindi utilizzeranno più frequentemente questo ultimo. Ciò contribuisce ad un'alta intensità di feromone come mostrato nella *fig.21.3*.

Nel momento in cui tutte le formiche avranno completato un percorso, ovvero quando saranno pervenute ad una soluzione, verrà aggiornata la quantità di feromone per la successiva iterazione dell'algoritmo. Il feromone infatti nel tempo evapora, questo spiega il fatto che i percorsi più lunghi avranno un potere attrattivo minore rispetto a quelli più brevi, poiché la

quantità di feromone sarà minore. *N* e *S* rappresentano rispettivamente il nido e la sorgente di cibo, mentre *x1, x2, x3, x4, x5* sono i vari nodi del grafo.

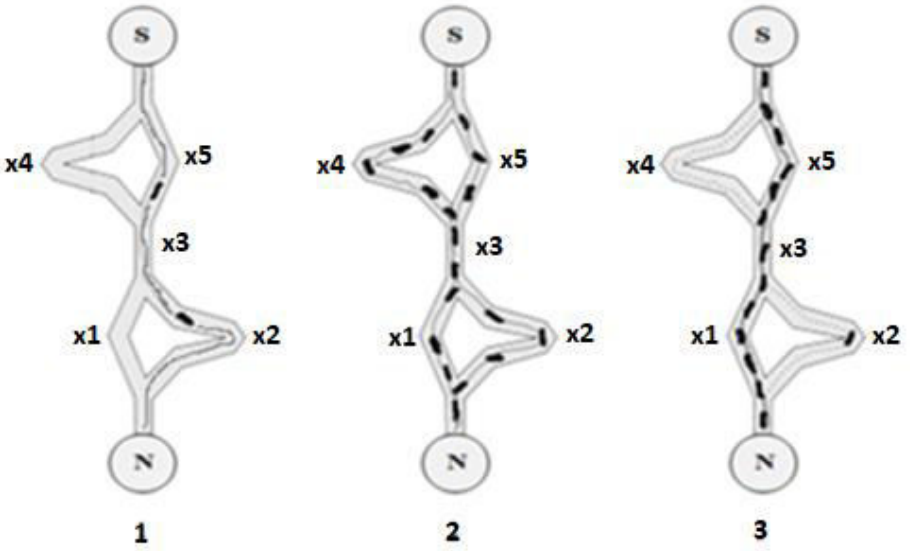


fig.21

Nell'ACO, ogni formica è considerata un agente artificiale che ripetutamente, crea delle soluzioni del problema, partendo dal nodo iniziale del grafo (*N*) e giungendo al nodo finale (*S*). L'agente artificiale quindi, ad ogni iterazione dell'algoritmo, percorre un tratto del grafo, muovendosi dal nodo in cui risulta essere collocato, ad un altro nodo adiacente connesso. Considerando di avere più nodi adiacenti connessi, l'agente ha una determinata probabilità di spostarsi in uno qualsiasi di questi ultimi. Questa probabilità oltre ad essere funzione della quantità di feromone presente tra i relativi nodi, dipende anche da un fattore di visibilità. Il valore di feromone incide sullo sfruttamento di una soluzione (percorso) precedentemente trovata da altri agenti. Il valore del fattore di visibilità, invece, incide sull'esplorazione di nuove possibili soluzioni, ossia percorsi non ancora individuati. Questo ultimo garantisce una convergenza non prematura nelle fasi iniziali dell'algoritmo. (Grigolo 2013)

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di formiche
- $\tau 0$ → valore di feromone iniziale;
- Pheromone parameter (α) → parametro che indica quanto il feromone incide sulla scelta del percorso;
- Heuristics parameter (β) → parametro che indica quanto il fattore di visibilità incide sulla scelta del percorso;

- Evaporation rate (ρ) → parametro che rappresenta la velocità di evaporazione del feromone.

L'ACO presenta diversi vantaggi, in particolare la capacità di ricerca parallela (parallelizzazione) garantisce ad esso un calcolo distribuito che evita una convergenza prematura. Quest'ultimo inoltre può essere utilizzato anche in applicazioni dinamiche. Tuttavia, l'ACO ha degli svantaggi, come una convergenza più lenta rispetto ad altri algoritmi di SI. Sebbene la convergenza sia garantita, il tempo di convergenza è incerto. Un altro importante demerito dell'ACO è la sua scarsa performance all'interno di problemi con grandi spazi di ricerca. (Ab Wahab, Nefti-Meziani, e Atyabi 2015)

In *fig.22* viene riportato lo pseudocodice dell'ACO.

Given:
-nA: number of ants.
-nI: number of iterations.
-iP: initial Pheromone.
Generate initial population of size nA (ants).
Initialize the pheromone trail and parameters.
Evaluate initial population according to the fitness function.
Find the best solution of the population.
While $(current\_iteration \leq nI)$
Do Until *each ant completely builds a solution*
local trial update.
End Do
Update the pheromone.
Determine the best global ant.
end While
Output the global best solution

fig.22

Campi di applicazione ACO

L'ACO è usato maggiormente per risolvere problemi riconducibili a percorsi su grafi, soprattutto per il "problema del commesso viaggiatore" TSP: (Valdez e Chaparro 2013) hanno utilizzato l'ACO per risolvere il TSP con elaborazione in parallelo. Applicazioni dell'Ant Colony Optimization possono essere riscontrare anche nei seguenti settori.

Vehicle Routing: (Fuellerer et al. 2009) si sono serviti dell'ACO per risolvere un problema nella logistica della distribuzione, ossia l'instradamento dei veicoli (caricati con le merci) lungo la rete stradale; Networking: (Kumar e Myilsamy 2013) hanno proposto un metodo Multi-

Target Tracking per la gestione ed il controllo della mobilità delle reti di sensori mobili, che utilizza come algoritmo l' Ant Colony Optimization. I risultati delle varie simulazione illustrano i vantaggi del metodo proposto rispetto ai metodi Single-Target Tracking sviluppati per le reti di sensori statici; Project Management: (Abdallah et al. 2009) hanno sfruttato l'ACO per la risoluzione ed il calcolo di reti CPM/PERT sia deterministiche che probabilistiche, ossia i due principali strumenti di project management. Il metodo proposto è studiato per un caso di gestione delle costruzioni. I risultati dimostrano che, a confronto con i metodi convenzionali, l'ACO può produrre buone soluzioni ottimali.

3.5.4 CS o CSA – Cuckoo Search Algorithm

L'algoritmo di ricerca del cuculo (CSA) è uno degli ultimi algoritmi metaeuristici ispirati alla natura, sviluppato nel 2009 da Xin-She Yang e Suash Deb. Questo algoritmo si basa sul parassitismo della covata di alcune specie di cuculo. I cuculi presentano una strategia di riproduzione molto aggressiva. Alcune specie depongono le uova in dei nidi comuni ed inoltre possono provvedere a rimuovere le uova di altre, al fine di aumentare la probabilità di schiusa delle proprie. Esistono tre tipi fondamentali di parassitismo da covata: parassitismo intraspecifico da covata, allevamento cooperativo e acquisizione del nido. Alcuni uccelli possono entrare in conflitto diretto con i cuculi che si sono intrusi nei loro nidi. Se un uccello scopre che le uova non sono di sua proprietà può sbarazzarsi di queste ultime, oppure potrebbe decidere di abbandonare il proprio nido e costruirne uno nuovo altrove. Alcune specie di cuculo si sono talmente evolute, che le femmine di tali specie riescono addirittura a deporre delle uova molto simili, sia nella forma che nei colori, alle uova presenti nel nido parassitato. Si sono quindi molto specializzate nel mimetismo. Questa caratteristica riduce la probabilità che le loro uova vengano "sfrattate" e quindi aumenta la loro riproduttività. (Gandomi, Yang, e Alavi 2013)

Oltre questo aspetto appena esposto, l'algoritmo CS presenta anche caratteristiche riconducibili ai "Lévy Flights". Vari studi hanno dimostrato infatti che il comportamento in volo di molti animali (come il cuculo) e insetti abbia caratteristiche tipiche dei "Levy Flights". (Reynolds e Frye 2007) hanno denotato il fatto che i moscerini della frutta (drosofile) esplorano il paesaggio utilizzando una serie di percorsi di volo rettilinei, caratterizzati da improvvise virate a 90°, che si riconducono ad un modello di ricerca libera in stile "Levy Flight".

L'algoritmo CS può essere descritto nel seguente modo, individuando le sue tre regole principali:

1) Ogni cuculo depone un uovo alla volta (ad ogni iterazione) e lo colloca arbitrariamente in un nido (possibile soluzione ottima);

2) Il numero di nidi disponibili è un parametro inizialmente fissato e l'uovo deposto da un cuculo può essere scoperto dall'uccello ospitante con una certa probabilità *Pa*. In questo caso l'uccello ospitante può liberarsi dell'uovo del cuculo, o semplicemente abbandonare il proprio nido e costruirne uno completamente nuovo;

3) I migliori nidi in cui depositare le uova (individuati tramite una funzione di fitness) verranno maggiormente utilizzati nelle successive iterazioni.

Dal punto di vista dell'implementazione, utilizzando un approccio basilare si può considerare che ogni nido rappresenti una soluzione (al problema di ottimizzazione) e in esso sia presente un solo uovo. Si considera inoltre che ogni cuculo possa deporre un solo uovo. In questo caso, non c'è distinzione tra uovo, nido o cuculo, poiché ogni nido corrisponde ad un uovo che rappresenta anche un cuculo. Ovviamente questo algoritmo può essere esteso al caso più complicato, in cui ogni nido ha più uova che rappresentano un insieme di soluzioni.

Quando si generano nuove soluzioni *x(t+1)*,ad esempio per il cuculo *i*, viene eseguito un Lévy Flight:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \alpha \oplus \text{Lévy}(\lambda) \qquad \textit{Lévy} \sim u = t^{-\lambda}, \ (1 < \lambda \leq 3)$$

L'equazione sopra riportata è essenzialmente un'equazione stocastica per una camminata (volata) casuale, dove α è la dimensione del passo. Nella maggior parte dei casi, si può inizializzare $\alpha = 1$. Questo valore diminuisce iterativamente permettendo alla popolazione di convergere verso la soluzione migliore. Inizialmente, come visto negli altri algoritmi, si predilige quindi l'esplorazione dello spazio di ricerca e successivamente lo sfruttamento delle migliori soluzioni individuate.

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di nidi

- Detection Probability (*Pa*)
- α

Anche il Cuckoo Search Algorithm (CSA) come l'Artifical Bee Colony (ABC) presenta pochi parametri da inizializzare, ciò gli garantisce un vantaggio competitivo rispetto i vari algoritmi di SI. Il CSA inoltre risulta essere facilmente implementabile e può essere ibridizzato in maniera semplice con altri algoritmi. Per di più garantisce una buona ricerca globale. D'altro canto la ricerca locale non risulta essere molto accurata e l'algoritmo per giunta ha una convergenza leggermente lenta. (Gandomi et al. 2013)

In *fig.23* viene riportato lo pseudocodice del CSA.

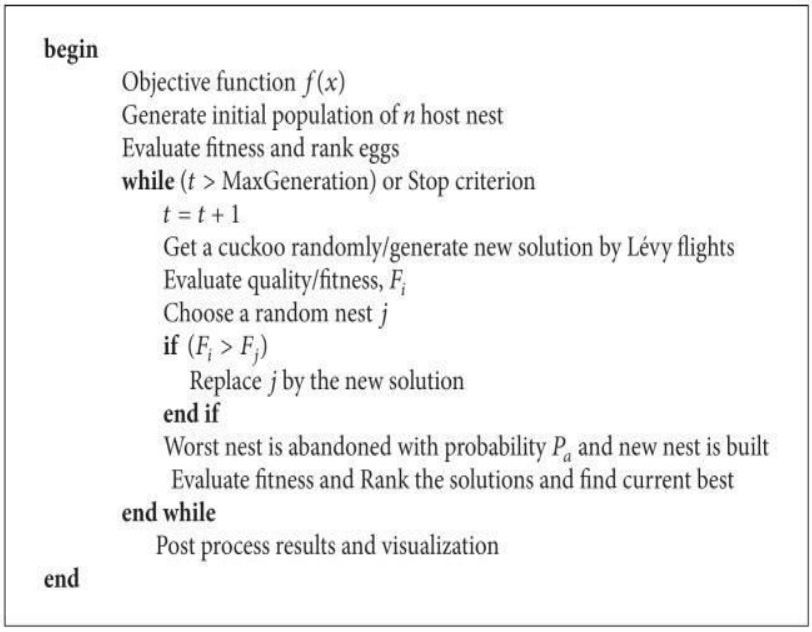


fig.23

Campi di applicazione CSA

Il Cuckoo Search Algorithm è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Embedded Systems: (Kumar e Chakarverty 2011) hanno utilizzato il CSA per risolvere i problemi che sorgono nell'allocazione e nella programmazione, al fine di garantire un'affidabile ottimizzazione di un sistema embedded. I risultati mostrano che il CSA fornisce soluzioni di alta qualità, poiché questo ultimo si basa costantemente su buone soluzioni trovate durante il processo di progettazione, inoltre è meno sensibile alle variazioni dei parametri e quindi riduce anche il tempo impiegato per la messa a punto del sistema. Conseguentemente riduce il time-

to-market; Manufacturing Problems: (Yildiz 2013) ha sfruttato il CSA per selezionare i parametri macchina ottimali nell'operazione di fresatura. Al fine di dimostrare l'efficacia del CSA, l'autore ha confrontato i risultati ottenuti con tale metodo, con quelli ottenuti utilizzando altri algoritmi di ottimizzazione ben noti come, ad esempio, l'ACO e il GA (Genetic Algorithm). I risultati dimostrano che il CSA è un approccio molto efficace e robusto per la risoluzione dei problemi di ottimizzazione delle lavorazioni; Engineering problems: (Bhargava, Fateen, e Bonilla-Petriciolet 2013) hanno implementato il CSA per risolvere alcuni calcoli termodinamici in applicazioni complesse dell'ingegneria chimica. Lo studio riporta l'applicazione del CSA e di una sua versione modificata. I risultati mostrano che la ricerca del cuculo offre una prestazione affidabile per risolvere questi tipi problemi termodinamici; Electronics application: (Moravej e Akhlaghi 2013) hanno utilizzato il CSA per effettuare un'allocazione ottimale delle unità elettriche di autoproduzione, al fine di migliorare il profilo di tensione e ridurre la perdita di potenza nella rete elettrica di distribuzione. Il CSA è stato confrontato con altri algoritmi come l'algoritmo genetico (GA) e l'ottimizzazione dello sciame di particelle (PSO). I risultati indicano una migliore prestazione del CSA, rispetto agli altri metodi, grazie al minor numero di parametri che devono essere regolati.

3.5.5 FA o FFA – FireFly Algorithm

L'algoritmo delle lucciole (FFA) è stato sviluppato da Xin-She Yang alla fine del 2007 e come indica il nome, si basa sul comportamento delle lucciole. La luce lampeggiante prodotta dalle lucciole è un processo di bioluminescenza e le vere funzioni di tali sistemi di segnalazione sono ancora oggi oggetto di dibattito. Tuttavia, due funzioni fondamentali di questi "flash" sono l'attrazione di partner per l'accoppiamento e l'attrazione di potenziali prede. L'intensità I della luce emessa da ciascuna lucciola, naturalmente decresce all'aumentare della distanza r da essa. Inoltre, anche l'aria assorbendo la luce contribuisce alla sua riduzione di intensità. Questi due fattori combinati rendono la maggior parte delle lucciole visibili ad una distanza limite.

La luce lampeggiante può essere formulata in modo tale da essere associata ad una funzione obiettivo da ottimizzare, il che rende possibile la formulazione del FireFly Algorithm. Nello sciame di lucciole ogni lucciola rappresenta quindi un agente di ricerca. (Yang 2009)
In sostanza, il FA è basato sulle seguenti tre regole idealizzate:

1) Le lucciole sono "unisex", in modo che ogni lucciola può essere attratta dalle altre, indipendentemente dal loro sesso;

2) L'attrattiva è proporzionale alla luminosità, ed entrambe diminuiscono all'aumentare della distanza. Considerando quindi due lucciole lampeggianti, la meno luminosa si sposterà verso la più luminosa. Se non ce n'è una prevalenza di luminosità di una particolare lucciola, allora si sposteranno in modo casuale;

3) La luminosità di una lucciola è influenzata dallo spazio di ricerca della funzione obiettivo.

Nel FFA ci sono due questioni importanti da considerare: la variazione dell'intensità della luce I e la formulazione dell'attrattività β. Per semplicità, si può sempre presumere che l'attrattiva di una lucciola sia determinata dalla sua luminosità, che a sua volta è associata alla funzione obiettivo. Poiché la luminosità di una lucciola, come visto precedentemente, è inversamente proporzionale alla distanza r, allora ciò sarà anche valido per quanto riguarda l'attrattività. Inoltre, l'intensità della luce viene assorbita anche nell'aria , quindi si deve permettere anche all'attrattività di variare con un coefficiente di assorbimento γ:

$$\beta = \beta_0 e^{-\gamma r^2}$$

dove β_0 è l'attrattiva quando $r = 0$. Il valore del parametro γ è di fondamentale importanza per determinare la velocità della convergenza dell'algoritmo. L'attrazione fornisce quindi meccanismi validi per lo sfruttamento (di soluzioni trovate, quelle più "luminose") e guida il sistema verso una rapida convergenza, ma con un'adeguata randomizzazione è anche possibile effettuare un certo livello di esplorazione. Per questo motivo, il movimento di una lucciola i che viene attratta da un'altra lucciola j più "attraente" (più luminosa) viene determinato con la seguente equazione:

$$x_i(\mathrm{t}+1) = x_i(t) + \beta_0 e^{-\gamma r_{ij}^2}[x_i(t) - x_j(t)] + \alpha_t \mu_i(t)$$

dove il secondo termine è dovuto all'attrazione. Il terzo termine rappresenta la randomizzazione, dove α_t è il parametro di randomizzazione e $\mu_i(t)$ è un vettore di numeri casuali tratti da una distribuzione gaussiana o da una distribuzione uniforme al tempo t. Se $\beta_0 = 0$, diventa una semplice passeggiata casuale senza nessuna "forza" di attrazione. D'altra parte, se $\gamma = 0$, il FFA si riduce ad una variante dell'ottimizzazione dello sciame di particelle (PSO). Inoltre $\mu_i(t)$ può essere facilmente esteso ad altre distribuzioni, come ad esempio i Lévy Flight visti precedentemente.

La distanza r_{ij} tra la lucciola *i* e la lucciola *j* è una distanza cartesiana:

$$r_{ij} = \|x_i - x_j\|$$

I parametri che devono essere inizializzati per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di lucciole
- α
- β (initial attractiveness)
- γ

Il FFA è in grado di scoprire contemporaneamente più soluzioni ottimali nello spazio di ricerca, ciò lo rende particolarmente adatto per applicazioni di ottimizzazione multimodale. Questa affermazione deriva dal fatto che le lucciole possono automaticamente suddividersi in pochi sottogruppi, perché l'attrazione locale è più forte di quella a lunga distanza. Questi sottogruppi inoltre lavorano in modo indipendente, ne consegue quindi che il FFA è anche adatto per l'implementazione parallela. Un altro vantaggio è il fatto di poter applicare tale algoritmo anche a problemi non lineari. Il FFA inoltre può essere ibridizzato in maniera semplice con altri algoritmi. Di contro però questo algoritmo ha una risoluzione non molto accurata e ci sono concrete possibilità di convergenza in ottimi locali. (Fister et al. 2013)

In *fig.24* viene riportato lo pseudocodice del FFA.

```
Function Firefly
Objective function f(X),      X=(x1, ...,xd)
Generate an initial population of fireflies
Evaluate light intensity Ii for firefly at Xi by using f (Xi) for all fireflies
Iteration=1;
While (Iteration<=max_iteration)
{
   For (i=1; i<=population _size; i++)
  {
      For (j=1; j<=population _size; j++)
      {
       If (Ii>Ij) Move firefly i towards j in d-dimension;
       Attractiveness varies with distance r via exp [- r];
       Evaluate light intensity of new solutions;
      }
  }
Update current best solution;
Iteration++;
}
Return current best solution;
End Function
```

fig.24

Campi di applicazione FFA

Il FFA è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Industrial Optimization: (Hu 2012) ha proposto il FFA per trovare una strategia ottimale in grado di risolvere il problema del consumo di energia nel trasporto ferroviario. I risultati dimostrano che questo algoritmo ha prestazioni migliori rispetto ad altri metodi. L'autore ha quindi realizzato un modello ad alta efficienza energetica per il controllo ottimale del funzionamento dei treni; Image Processing: (Horng e Jiang 2010) hanno utilizzato il FFA per sviluppare un nuovo algoritmo di quantizzazione vettoriale di un immagine (tecnica utilizzata per la compressione/codifica digitale delle immagini). Questo nuovo metodo è stato confrontato con altri tre. I risultati sperimentali hanno mostrato che tale metodo fornisce immagini "ricostruite" con qualità più alta di quelle generate dagli altri; Clustering: (Senthilnath, Omkar, e Mani 2011) hanno applicato il FFA per eseguire il raggruppamento di oggetti, in un insieme di dati, secondo i valori dei loro attributi. Le prestazioni del FFA sono state poi comparate con altri due algoritmi nature-ispired (ABC e PSO) e altri nove metodi utilizzati in letteratura. Dai risultati ottenuti, gli autori concludono che il FFA è un metodo efficiente, affidabile e robusto, che può essere quindi applicato per generare con successo cluster ottimali; Scheduling: (Udaiyakumar e Chandrasekaran 2014) hanno applicato il FFA al problema del Job Shop Scheduling al fine di minimizzare il Makespan, ovvero il tempo che passa da quando viene rilasciato il job, fino all'istante di completamento dell'ultimo pezzo in produzione nell'ultima macchina. Più il Makespan è basso e più le prestazioni del sistema produttivo saranno migliori.

3.5.6 GWO – Grey Wolf Optimizer

(Mirjalili, Mirjalili, e Lewis 2014) nel 2014 hanno introdotto il Grey Wolf Optimizer (GWO) che imita la gerarchia della leadership e il meccanismo di caccia dei branchi di lupi grigi in natura. L'algoritmo divide gli agenti (lupi grigi) nelle loro diverse categorie gerarchiche denominate rispettivamente alfa (α), beta (β), delta (δ) e omega (ω) (*fig.25*).

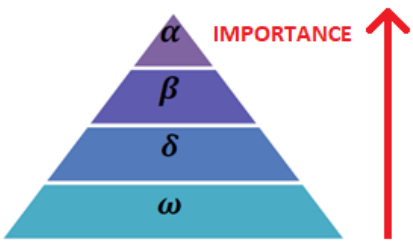


fig.25

Ogni gerarchia ha ruoli diversi per trovare le soluzioni, che in questo caso corrispondono alle prede. I leader dei branchi sono i lupi alfa. L'alfa è il principale responsabile delle decisioni inerenti alla caccia, al luogo dove dormire e così via. Il lupo alfa è il lupo dominante che impartisce gli ordini ,che vengono poi seguiti dal branco. Il secondo livello nella gerarchia sono i lupi beta. I beta sono lupi subordinati che aiutano l'alfa nel processo decisionale o in altre attività del branco. Il lupo beta deve rispettare l'alfa, ma inoltre comanda altri lupi di livello inferiore. Il lupo grigio di livello più basso è l'omega. L'omega svolge il ruolo di capro espiatorio. Questo aiuta a soddisfare l'intero branco e a mantenere la struttura dominante. I lupi omega devono sempre sottomettersi a tutti gli altri lupi dominanti. 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 degli omega. Se un lupo non è alfa, beta o omega, viene chiamato subordinato (o delta). I lupi delta devono sottomettersi agli alfa e ai beta, ma dominano gli omega. Scout, cacciatori ecc. appartengono a questa categoria. Gli scout hanno la responsabilità di sorvegliare i confini del territorio e di avvertire il branco in caso di pericolo. I cacciatori aiutano gli alfa e i beta nella caccia alle prede e forniscono cibo al branco.

Al fine di modellare matematicamente la gerarchia sociale dei lupi nella progettazione del GWO, si considera quindi α la soluzione più adatta (ottimale). Di conseguenza, la seconda e la terza soluzione migliore sono denominate rispettivamente β e δ. Le restanti soluzioni candidate sono assunte come ω.

Nell'algoritmo GWO la caccia (ottimizzazione) è guidata da α, β e δ. I lupi ω seguono questi tre precedenti categorie di lupi. Vengono quindi implementate le fasi principali di caccia del branco di lupi:

- ricercare ed inseguire la preda;
- accerchiare e molestare la preda fino a quando non smette di muoversi;
- attaccare la preda.

Fase di Accerchiamento della preda:

Dopo aver individuato una possibile preda, i lupi iniziano ad accerchiala per poi successivamente passare all'attacco. Per modellare matematicamente il comportamento di accerchiamento vengono proposte le seguenti equazioni:

$$D = \left| C \,.\, x_p(t) - x(t) \right|, \qquad x(t+1) = x_p(t) - A \,.\, D$$

dove il vettore *D* rappresenta la differenza tra la posizione della preda e del predatore (lupo) , *t* denota l'iterazione corrente, il vettore x_p specifica la posizione della preda e il vettore *x* indica la posizione del lupo. I valori dei vettori *A* e *C* possono essere determinati attraverso le seguenti equazioni:

$$A = 2a \,.\, r_1 \; - \; a \,, \qquad\qquad C = 2 \,.\, r_2$$

dove le componenti del vettore *a* vengono diminuite linearmente da 2 a 0 nel corso delle iterazioni e r_1, r_2 sono dei vettori casuali con valori compresi nell'intervallo [0,1], che permettono ai lupi di raggiungere qualsiasi posizione tra i punti illustrati in *fig.26*.

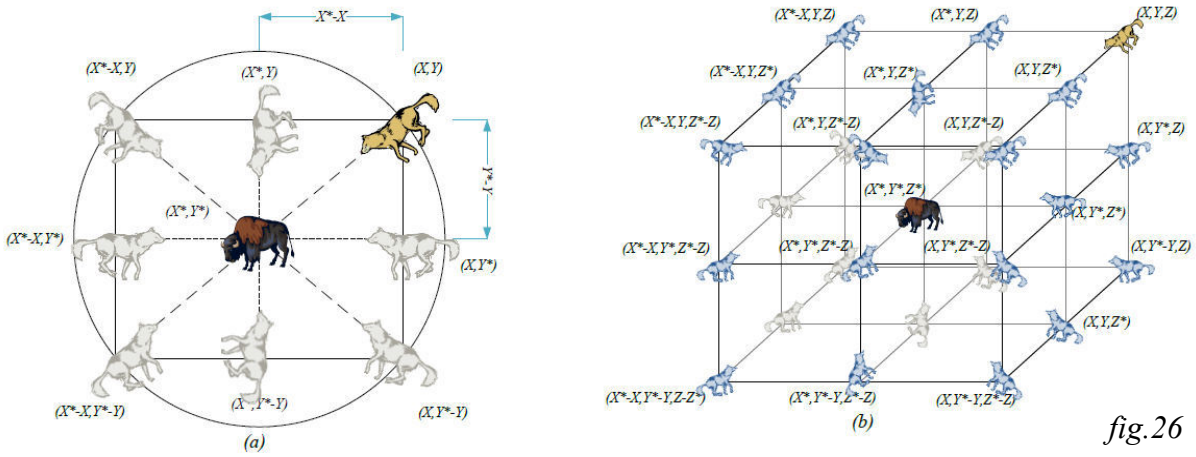


fig.26

Come si può vedere in *fig.26a*, un lupo nella posizione (X,Y) può aggiornare la sua posizione, secondo la posizione della preda (X*,Y*). I possibili aggiornamenti di posizione di un lupo nello spazio 3D, sono invece rappresentate nella *fig.26b*. Lo stesso concetto può essere quindi esteso ad uno spazio di ricerca con *n* dimensioni, dove i lupi si muoveranno in iper-cubi (o ipersfere) intorno alla migliore soluzione trovata.

Fase di Caccia della preda individuata:

Al fine di simulare matematicamente il comportamento di caccia dei lupi, si suppone che alfa (migliore soluzione candidata) beta e delta abbiano una migliore conoscenza della posizione potenziale della preda. Pertanto, si considerano le prime tre migliori soluzioni ottenute e si obbligano gli altri agenti di ricerca (lupi omega) ad aggiornare le loro posizioni in base alla posizione del miglior agente di ricerca.

Fase di Attacco (sfruttamento):

Come detto sopra, i lupi finiscono la caccia attaccando la preda quando questa smette di muoversi. Al fine di modellare matematicamente l'avvicinamento alla preda si diminuisce il valore del vettore *a*. Di conseguenza anche i valori del vettore *A* diminuiranno. Quando *A* presenta valori compresi in [-1,1], ossia quando $|A|<1$ i lupi iniziano la fase di attacco spostandosi sempre di più verso la preda (*fig.27a*).

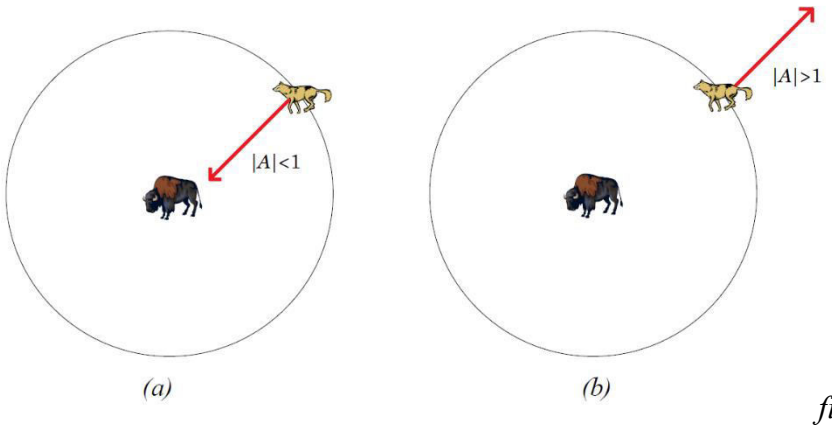


fig.27

Fase di Ricerca delle prede (esplorazione):

I lupi ricercano la preda soprattutto in base alla posizione di alfa, beta e delta. In questa fase di ricerca (esplorazione) i lupi si allontanano l'uno dall'altro, al fine di individuare le diverse posizioni delle prede (soluzioni). Per modellare matematicamente la divergenza, il vettore *A* assume valori maggiori di 1 o minori di -1 e obbliga l'agente di ricerca a divergere dalla preda. Questo enfatizza l'esplorazione e permette all'algoritmo GWO di cercare globalmente, al fine di trovare una preda migliore. L'esplorazione è facilitata anche dal vettore *C* se presenta valori maggiori di 1. Il vettore *C* può fornire valori casuali ad ogni iterazione dell'algoritmo, al fine di enfatizzare l'esplorazione non solo durante le iterazioni iniziali ma anche in quelle finali. Al contrario del vettore *A* infatti, il vettore *C* non diminuisce linearmente. Questo componente inoltre è molto utile in caso di "stagnazione" in ottimi locali, soprattutto nelle iterazioni finali.

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di lupi
- *a*
- *A*
- *C*

Il GWO presenta il vantaggio di avere pochi parametri da inizializzare e di essere un algoritmo flessibile. Questo ultimo infatti può adattarsi a diversi problemi pratici di carattere ingegneristico. Il GWO per di più, può essere facilmente implementabile e grazie alla sua struttura gerarchica garantisce un'alta precisione di ricerca. Di contro però, essendo stato introdotto recentemente rispetto ad altri algoritmi di SI, le attività di ricerca e sviluppo per tale algoritmo non sono ancora ad una fase evoluta. Come enunciato precedentemente il GWO ha una forte capacità di esplorazione e può quindi evitare la convergenza in ottimi locali. Questa caratteristica però, può portare l'algoritmo ad avere una lenta convergenza. (Wang e Li 2019)

In *fig.28* viene riportato lo pseudocodice del GWO.

```
Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    t=t+1
end while
return X_α
```

fig.28

Campi di applicazione GWO

Pur essendo stato introdotto recentemente, il Grey Wolf Optimizer è stato utilizzato in diversi campi applicativi. Di seguito ne sono riportati alcuni esempi.

Control of Systems: (Das et al. 2015) hanno sperimentato il GWO per ottimizzare i parametri di un controllore PID, utilizzato per il controllo della velocità di un sistema di motori in corrente continua. Il miglior set di parametri ottenuto dall'ottimizzazione è stato successivamente impostato nel PID. Gli autori hanno poi fornito un ingresso a gradino al motore DC, per ottenere le specifiche di risposta al transitorio, come il tempo di salita, il tempo di assestamento e il massimo over-shoot. Questi risultati sono stati poi confrontati con i risultati di altre tecniche convenzionali, al fine di individuare il metodo più efficiente, per ottenere la migliore risposta

ai transitori. Dal lavoro svolto è risultato che l'ottimizzazione del PID, utilizzando l'algoritmo GWO, fornisce la migliore serie di specifiche di risposta ai transitori rispetto agli altri metodi; Scheduling: (Komaki e Kayvanfar 2015) hanno proposto l'applicazione del GWO per programmare la sequenza ottimale di lavorazione e di assemblaggio, al fine di minimizzare il tempo di completamento del lavoro. I risultati ottenuti con tale algoritmo sono stati confrontati poi con altri metodi. Tale confronto ha rivelato che il GWO garantiva prestazioni migliori; Networking: (Nguyen, Thom, e Dao 2017) hanno utilizzato un GWO multi-obiettivo per risolvere il problema della localizzazione dei nodi, in una rete di sensori wireless. Nel modello di localizzazione sono stati considerati vari vincoli, tra cui il vincolo della distanza spaziale tra i nodi e il vincolo della topologia. I risultati delle simulazioni mostrano notevoli miglioramenti in termini di precisione della localizzazione e di tasso di convergenza alla soluzione ottimale, rispetto a quelli ottenuti con gli altri metodi; Environmental modeling: (Song et al. 2015) hanno utilizzato il GWO per stimare dei parametri delle Onde di Rayleigh (un tipo onde elastiche superficiali).

3.5.7 BA – Bat Algorithm

L'algoritmo del pipistrello "Bat Algorithm" (BA), proposto da Xin-She Yang nel 2010, è un ulteriore algoritmo basato sulla Swarm Intelligence e ispirato al comportamento di eco localizzazione dei pipistrelli. (Yang e He 2013)

Tra tutte le varie specie di pipistrello, i microchirotteri (micro-bats) usano l'eco localizzazione in maniera più estensiva. La maggior parte dei microchirotteri sono insettivori e durante il volo e la caccia, emettono nell'ambiente alcuni brevi impulsi ultrasonori della durata di qualche millesimo di secondo (fino a circa 8-10 ms). Successivamente ricevono ed ascoltano i corrispettivi echi degli impulsi che rimbalzano negli oggetti circostanti. I micro-bats analizzando il ritardo temporale tra l'emissione dell'impulso e il rilevamento dell'eco e le variazioni di intensità degli echi, sono in grado di costruire uno scenario tridimensionale dell'ambiente circostante anche nel buio più completo. Possono rilevare la distanza, l'orientamento e il tipo di ostacolo o di preda. Inoltre sono in grado di rilevare anche la velocità di movimento di quest'ultima (*fig.29*).

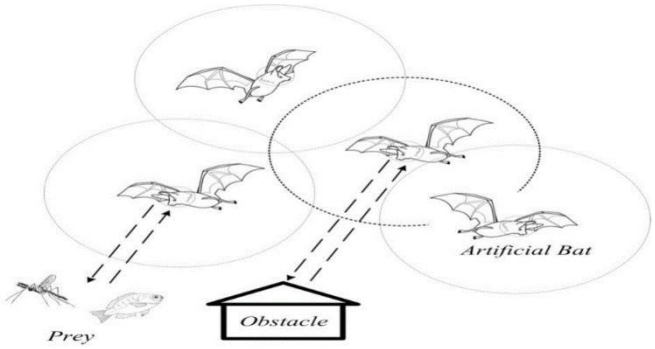


fig.29

Tipicamente i micro-bats possono emettere da 10 a 20 impulsi ultrasonori ogni secondo. La velocità di emissione degli impulsi però, può essere accelerata fino a circa 200 impulsi al secondo durante la fase di attacco della preda. La frequenza tipica degli impulsi ultrasonori varia in un range da 25kHz a 150kHz, mentre la corrispettiva lunghezza d'onda varia tra i 2mm e i 14mm. È interessante notare che queste lunghezze d'onda sono dello stesso ordine di dimensione delle prede.

Il comportamento di eco localizzazione dei micro-bats può essere formulato in modo tale da essere associato alla funzione obiettivo da ottimizzare. Come tutti gli algoritmi basati sulla SI, il meccanismo di ricerca di tale algoritmo è governato da due componenti cruciali: lo sfruttamento e l'esplorazione. Xin-She Yang ha sviluppato il BA idealizzando le tre seguenti regole:

1) Tutti i pipistrelli usano l'eco localizzazione per percepire la distanza di ostacoli e prede, riuscendo inoltre anche a distinguerli;

2) I pipistrelli volano in modo casuale con velocità v_i in posizione x_i, ed emettono impulsi ultrasonori con frequenza f (o lunghezza d'onda λ) e intensità A_0 per cercare la preda. Possono inoltre regolare automaticamente la lunghezza d'onda (o frequenza) e il tasso di emissione degli impulsi $r \in [0,1]$, a seconda della vicinanza del loro bersaglio;

3) Anche se l'intensità sonora può variare in molti modi, si assume che quest'ultima vari tra un valore massimo (positivo) A_0 ed un valore minimo A_{min}.

Ogni pipistrello i, all'iterazione t, è associato ad una velocità $v_i(t)$ e ad una posizione $x_i(t)$, nello spazio delle soluzioni. Tra tutte le posizioni della popolazione degli n pipistrelli, la x^*

corrisponde alla migliore soluzione trovata all'iterazione *t*. Di seguito vengono definite le equazioni di aggiornamento delle velocità e delle posizioni all'iterazione *t+1*:

$$f_i = f_{min} + (f_{max} - f_{min})\beta$$
$$v_i(t+1) = v_i(t) + [x_i(t) - x^*]fi$$
$$x_i(t+1) = x_i(t) + v_i(t+1)$$

dove $\beta \in [0,1]$ è un vettore casuale. Come accennato precedentemente, per l'implementazione dell'algoritmo si possono usare le lunghezze d'onda o le frequenze f_{min} e f_{max}. Queste ultime vengono definite in base alla dimensione del dominio del problema di interesse.

Inizialmente, ad ogni pipistrello è assegnata casualmente una frequenza f_i ; questa frequenza, responsabile del movimento del pipistrello, deve garantire una combinazione equilibrata di esplorazione di nuove posizioni (soluzioni) e sfruttamento di quelle già individuate. Al fine di fornire un meccanismo efficace per controllare l'esplorazione e passare alla fase di sfruttamento quando necessario, durante le iterazioni si varia l'intensità A_i e il tasso di emissione degli impulsi r_i. L'intensità di solito diminuisce una volta che un pipistrello ha trovato la sua preda, mentre il tasso di emissione di impulsi aumenta. Il valore dell'intensità inoltre può essere selezionato nell' intervallo $[A_{min}, A_{max}]$; assumendo $A_{min} = 0$ significa che un pipistrello ha appena trovato la preda. Il tasso di emissione degli impulsi e l'intensità vengono aggiornati solo se le nuove soluzioni sono migliori delle precedenti, il che significa che i pipistrelli si stanno muovendo verso la soluzione ottimale.

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di pipistrelli
- A_i
- r_i
- f_{min}
- f_{max}

Il BA ha il vantaggio di essere semplice e flessibile. Il BA infatti, è facile da implementare, e può essere usato per risolvere un'ampia gamma di problemi. Di seguito vedremo alcuni campi applicativi in cui è stato utilizzato. Uno dei vantaggi chiave del BA è la capacità di zoomare

automaticamente (Automatic Zooming) in una regione in cui sono state trovate soluzioni promettenti. Questo zoom è accompagnato dal passaggio automatico dai movimenti esplorativi, allo sfruttamento intensivo locale. Di conseguenza, il BA, rispetto ad altri algoritmi può fornire una convergenza molto rapida anche nelle prime fasi delle iterazioni. Questo lo rende un algoritmo efficiente per applicazioni dove è necessaria una soluzione rapida. Tuttavia, se si permette all'algoritmo di passare alla fase di sfruttamento troppo rapidamente (variando A e r troppo rapidamente), si può incorrere in una "stagnazione" in soluzioni sub-ottimali o ottimi locali. Anche il BA può quindi presentare il problema della convergenza prematura. (Yang e He 2013)

In *fig.30* viene riportato lo pseudocodice del BA.

Initialize the bat population X_i (i=1,2,…..,n) and V_i
Define pulse frequency F_i
Initialize pulse rate r_i and the loudness A_i
While (t < Max number of iterations)
Generate new solutions by adjusting frequency,
Updating velocities and positions
If (rand > r_i)
Select a solution among the best solutions randomly
Generate a local solution around the selected best solution
End if
Generate a new solution by flying randomly
If($rand < A_i$ & $f(x_i) < f(Gbest)$)
Accept the new solutions
Increase r_i and reduce A_i
End if
Rank the bats and find the current *Gbest*
End while

fig.30

Campi di applicazione BA

Il BA è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Power Systems: (Chevuru e Martheswar 2014) hanno dimostrato che implementando il BA, è possibile ottimizzare il funzionamento di impianti di generazione di energia. L'obiettivo finale era quello di far produrre a tali impianti, energia al minor costo, per soddisfare in modo affidabile le richieste dei carichi collegati; Designing: (Yang, Karamanoglu, e Fong 2012) hanno utilizzato il BA per risolvere i problemi di ottimizzazione della topologia in applicazioni microelettroniche, raggiungendo così un'efficiente progettazione della configurazione geometrica con una quantità ridotta di materiali; Image Processing: (Akhtar, Ahmad, e Abdel-

Rahman 2012) hanno utilizzato il BA per la stima della posa e del tracciamento del movimento umano, a partire da dati in formato video, registrati in diversi punti di vista. Le prestazioni del BA sono state confrontate con quelle di altri algoritmi come il PSO, utilizzando lo stesso set di dati. Tali analisi hanno evidenziato che il BA è risultato essere più efficace degli altri algoritmi.

3.5.8 AFSA – Artificial Fish Swarm Algorithm

L'AFSA è un algoritmo di SI che si ispira al movimento collettivo dei banchi di pesci e ai loro vari comportamenti sociali. I pesci muovendosi in gruppo dimostrano comportamenti sociali intelligenti. La ricerca di cibo, l'immigrazione e la gestione dei pericoli infatti, avvengono tutti in una forma sociale, tramite varie interazioni tra tutti i pesci del banco. I pesci, in banchi, possono reagire rapidamente ai cambiamenti di direzione e velocità dei vicini. Sono in grado quindi passare da una configurazione all'altra come se fossero un'unica unità. Tale algoritmo è stato sviluppato per la prima volta nel 2002 da Li Xiaolei. (Neshat et al. 2012)

Nell' algoritmo AFSA vengono considerati gli "Artificial Fishes" (AFs), che rappresentano entità fittizie di pesci reali. L'ambiente in cui vive l'AF è principalmente lo spazio della soluzione. Il comportamento successivo di un AF dipende dal suo stato attuale e dagli stati degli AFs vicini. Attraverso la propria vista, un AF riesce a percepire l'ambiente circostante ad esso (*fig.31*). *X* è il suo stato attuale, *Visual* è la distanza visiva, e *Xv* rappresenta la posizione visiva in cui l'AF tenderebbe a spostarsi. Se lo stato nella posizione visiva è migliore dello stato attuale, l'AF avanza di un "passo" (*Step*) nella direzione di *Xv* e arriva allo stato successivo *Xnext*; altrimenti continua ad effettuare un'ispezione visiva dell'ambiente circostante per un certo numero di volte *try_number*. Maggiore è il numero di ispezioni visive, maggiore sarà la conoscenza dei vari stati circostanti.

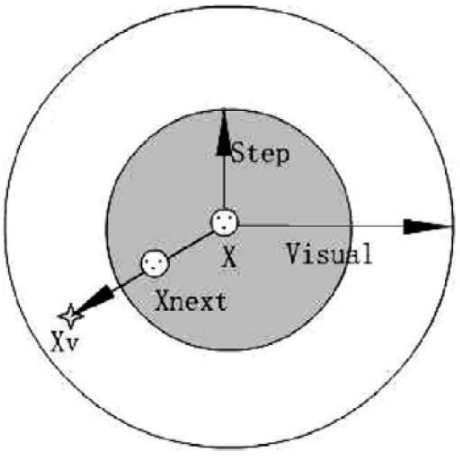


fig.31

Il processo di movimento descritto precedentemente viene formulato nel seguente modo:

$$x_i^v = x_i + Visual.rand() \qquad i \in (0, n]$$

$$X_{next} = X + \frac{X_v - X}{\|X_v - X\|}.Step.rand()$$

dove *rand( )* è un vettore casuale di numeri compresi tra 0 e 1, *Step* è la lunghezza del passo e x_i è la componente *i* dello stato *X n*-dimensionale. Le variabili dell'AFSA sono: *X*, *Step*, *Visual*, *try_number* e δ che rappresenta il fattore di affollamento.

Funzioni base dell'AFSA

I pesci di solito rimangono nel luogo in cui c'è un elevata presenza di cibo. Nell'algoritmo quindi si simula il comportamento dei pesci in base a questa caratteristica al fine di trovare la soluzione ottima globale. I comportamenti (le funzioni) di base dell'AF vengono definiti di seguito:

1) AF_Prey

   Rappresenta il modo in cui il pesce va a caccia di cibo; generalmente il pesce percepisce la concentrazione di cibo nell'acqua e determina il suo movimento in base ad essa. Si consideri X_j lo stato attuale dell'AF e si scelga uno stato X_k in modo casuale nella sua distanza visiva. *Y* è la concentrazione di cibo (valore della funzione obiettivo). Se $Y_j < Y_k$ allora l'AF avanza di uno *Step* nella direzione dello stato X_k (in un problema di massimizzazione). In caso contrario si seleziona un nuovo stato X_{k_1} sempre in modo casuale e si valuta se questo ultimo soddisfa la condizione anteriore. Se tale condizione non è soddisfatta entro un certo numero di tentativi *try_number*, l'AF si sposta di uno *Step* in maniera casuale.

2) AF_Swarm

   I pesci nel processo di spostamento, come detto precedentemente si muovono in banchi al fine di garantire l'esistenza di una colonia ed evitare pericoli. Sia X_i sia lo stato attuale di un AF, X_c la posizione centrale della zona del banco in cui l'AF si trova insieme ai "suoi" AF vicini. Sia m_f il numero degli AF vicini: un AF in posizione X_j è vicino dell'AF in posizione X_i se la distanza $d_{ij} < Visual$. Sia *m* il numero totale dei pesci nel banco. Allora se $Y_c > Y_i$ e $m_f/m < \delta$, il che significa che la posizione X_c ha una maggiore concentrazione di cibo (valore di funzione fitness più alto) e non è molto affollata, allora l'AF in posizione X_i si sposta di uno *Step* verso la posizione X_c. In caso contrario, l'AF

esegue il comportamento predatorio (AF_Prey). Il fattore di affollamento δ viene usato quindi per limitare la dimensione dei banchi.

3) AF_Follow

Nel processo di spostamento dello sciame di pesci, quando un singolo pesce o più pesci individuano la posizione di una fonte di cibo, i pesci vicini (del stessa zona) seguono le tracce dei precedenti e raggiungono rapidamente tale posizione. Sia X_i lo stato attuale di un AF, e sia X_j la posizione ispezionata visivamente dall'AF_i, in cui si trova invece l'AF_j ($d_{ij} < Visual$). Allora se $Y_j > Y_i$ e $m_f/m < \delta$, il che significa che la posizione X_j ha una maggiore concentrazione di cibo (valore di funzione fitness più alto) e non è molto affollata, allora l'AF_i si sposta di uno *Step* verso la posizione X_j. Altrimenti, esegue il comportamento predatorio (AF_Prey).

4) AF_Move

I pesci nuotano in modo casuale, o sono alla ricerca di cibo o di altri pesci "compagni". Si seleziona uno stato (posizione) casuale presente all'interno del raggio della *Visual* dell'AF e successivamente questo ultimo si sposta verso tale posizione.

5) AF_Leap

Quando un gran numero di pesci si arresta in una posizione, il risultato del comportamento degli altri pesci del banco tende ad essere lo stesso (raggiungono tale posizione). La differenza dei valori delle varie Y (concentrazione di cibo) per ogni posizione X, tende man mano a ridursi. La funzione obiettivo quindi durante queste iterazioni rimane pressoché costante, oppure varia di pochissimo. In questi casi si può presentare la convergenza in ottimi locali; occorre quindi cambiare in modo casuale le posizioni di alcuni AFs (AF_{some}), in modo tale che nelle successive iterazioni siano in grado di "saltare fuori" dalla posizione ottima locale in cui si trovano e valutare altre posizioni al fine di trovare l'ottimo globale.

$$X_{some}(t+1) = X_{some}(t) + \beta.Visual.rand()$$

dove β è un parametro che fa sì che alcuni AFs abbiano degli spostamenti anomali.

I parametri da inizializzare per l'implementazione di questo algoritmo sono i seguenti:

- nr. di iterazioni
- nr. di pesci
- δ
- *Step*
- *try_number*
- *Visual*

L'AFSA presenta vari vantaggi tra cui una buona capacità di ricerca globale e la capacità di ricerca parallela distribuita. Come la maggior parte degli algoritmi Swarm ,risulta inoltre essere flessibile ed in grado di adattarsi quindi a più classi di problemi differenti. Per di più presenta anche un'elevata velocità di convergenza. Tuttavia presenta anche degli svantaggi come la possibilità di convergere prematuramente in ottimi locali e la mancanza di equilibrio tra la ricerca globale e la ricerca locale. (Neshat et al. 2012)

In *fig.32* viene riportato lo pseudocodice dell'AFSA.

Standard AFSA

foreach *AF* i
  initialize x_i
endfor
bulletin = $arg\ \min_{X_i} f\ (X_i)$
Repeat
  foreach *AF i*
    Perform *swarm behavior* on $X_i(t)$ and obtain $X_{i,swarm}$
    Perform Follow behavior on $X_i(t)$ and obtain $X_{i,follow}$
    if $f(X_{i,swarm}) \geq f(X_{i,follow})$ **then**
      $X_i(t+1) = X_{i,swarm}$;
    else
      $X_i(t+1) = X_{i,follow}$;
    endif
  endfor
  if $f(X_{Best\text{-}AF}) \leq f(bulletin)$ **then**
    $bulletin = X_{Best\text{-}AF}$;
  endif
until stopping criterion is met

fig.32

Campi di applicazione AFSA

L'AFSA è utilizzato in vari campi applicativi, di seguito ne sono riportati alcuni esempi.

Path Planning: (Qianzhi Ma e Xiujuan Lei 2010) hanno proposto l'AFSA per risolvere il problema della pianificazione del percorso dei veicoli aerei da combattimento senza equipaggio

(UCA V) nell'ambito delle minacce radar 2D. I risultati delle simulazioni hanno mostrato che il metodo AFSA proposto è in grado di trovare un percorso con distanza di volo più breve ed evitare eventuali minacce in modo efficace. Successivamente gli autori hanno fatto anche un confronto con i risultati ottenuti utilizzando altri tipi di algoritmi. L'AFSA è risultato essere il metodo più adatto a risolvere questo tipo di problema; Networking: (Liu et al. 2009) hanno utilizzato l'AFSA per creare un metodo di ottimizzazione e semplificazione della funzionalità di rete, con rilevamento di eventuali intrusioni. Tale metodo ha stabilito un modello matematico volto ad ottenere una maggiore velocità di rilevamento e una minore percentuale di falsi positivi. I risultati sperimentali hanno dimostrato che questa ottimizzazione e semplificazione può ridurre del 40% il tempo necessario al rilevamento delle intrusioni; Scheduling: (Taixiong Zheng e Jiongqiu Li 2010) hanno sviluppato l'AFSA per risolvere il problema dell'assegnazione e della programmazione della sequenza ottimale dei compiti da far svolgere ad un sistema multi-robot. Tutto ciò al fine di minimizzare il tempo di elaborazione dei robot. Tramite questo algoritmo ad ogni robot viene assegnata la giusta quantità di compiti e si ottiene lo schema di assegnazione dei compiti ottimale. Per convalidare l'efficacia dell'approccio proposto sono stati fatti esperimenti e simulazioni che hanno poi riportato risultati soddisfacenti; Signal Processing: (Mingyan Jiang e Dongfeng Yuan 2005) hanno utilizzato l'AFSA per implementare un metodo ottimale di elaborazione dei segnali, al fine di ottenere la soglia di riduzione ottimale del rumore. L'obiettivo di tale studio consisteva nel ricostruire un segnale originario, partendo dallo stesso, affetto però da disturbi/rumori. I risultati sperimentali hanno dimostrato che l'algoritmo proposto, presentava prestazioni migliori rispetto ad un altro metodo convenzionale.

Capitolo 4

Modello Digitale dell'eiettore : Simulazioni con Matlab e risultati

Nell' ultimo capitolo verranno spiegate le simulazioni svolte con il software Matlab, al fine di realizzare il modello digitale dell'eiettore, rappresentato da un'equazione parametrica. Saranno, in primo luogo, selezionati ed implementati alcuni degli algoritmi di Swarm Intelligence tra gli 8 proposti e verranno scelte le tipologie di dati sperimentali da utilizzare appunto per la realizzazione del modello. Successivamente saranno ipotizzate e testate 3 diverse equazioni parametriche, con lo scopo di individuare la migliore, ossia quella che meglio rappresenta il comportamento del dispositivo preso in esame.

4.1 Obiettivo e preparazione alle simulazioni

Al fine di effettuare una corretta identificazione parametrica dell'eiettore, come indicato nel precedente capitolo, ci avvaliamo di algoritmi di Swarm Intelligence. In particolare, l'obiettivo sarà quello di formulare e testare virtualmente (tramite l'utilizzo del software Matlab) diverse equazioni parametriche, al fine di individuare la migliore tra quelle proposte, ossia quella che rappresenta più fedelmente il comportamento dell'eiettore stesso. Tale equazione rappresenterà quindi il modello matematico del dispositivo preso in esame.

Un'equazione parametrica può essere rappresentata dalla seguente formula:

$$a_1 x_1 + a_2 x_2 + a_3 x_3 = y$$

In questo caso si tratta di un'equazione di 1° grado, dove i termini a_1,a_2 e a_3 risultano essere 3 parametri, mentre x_1, x_2, x_3 e y rappresentano le variabili. Considerando di adottare ad esempio tale equazione come modello matematico dell'eiettore, si potrebbero considerare le variabili x come ingressi e la variabile y come uscita.

Come detto in precedenza, non conoscendo le equazioni del modello fisico dell'eiettore, sarà necessario eseguire un'identificazione "Black-Box", sfruttando come ingressi e come uscite, dei dati precedentemente acquisiti tramite misurazioni sperimentali, effettuate con i sensori collocati nell'impianto dove è presente il dispositivo preso in esame. Dato l'elevato numero di misurazioni effettuate, a scopo puramente indicativo, viene riportata in *fig.33* solo una sezione della tabella dove sono stati riportati i dati.

MISURAZIONI	PLiq	PDiffusore	PSerb	QLiq	QGas	D1	D2	PGas
1	10,33683	1,067455	1,098099	13,64139	46,15385	11	29	0,979393
2	10,33683	1,367104	1,421508	13,65896	19,56522	11	29	0,979393
3	10,33683	1,802706	1,840691	13,7117	15,38462	11	29	0,979393
4	10,33683	2,001085	2,046118	13,69412	13,77551	11	29	0,979393
5	10,33683	1,085489	1,099487	15,50477	57,69231	11	29	0,991187
6	10,33683	1,365717	1,424284	15,46961	43,47826	11	29	0,991187
7	10,33683	1,777735	1,837915	15,46961	26,62722	11	29	0,991187
8	10,33683	1,995536	2,058611	15,46961	24,12869	11	29	0,991187
9	10,33683	1,000214	1,000214	15,59267	79,717525	11	29	0,975926
10	10,33683	1,088143	1,106427	15,59267	62,71777	11	29	0,975926
11	10,33683	1,21054	1,21054	15,59267	59,089046	11	29	0,975926
12	10,33683	1,310548	1,310548	15,59267	53,249903	11	29	0,975926
13	10,33683	1,34907	1,40624	15,59267	50,84745	11	29	0,975926
14	10,33683	1,5215	1,5215	15,59267	44,583201	11	29	0,975926
15	10,33683	1,66254	1,66254	15,59267	39,176271	11	29	0,975926
16	10,33683	1,776348	1,837915	15,59267	32,02847	11	29	0,975926
17	10,33683	1,978889	2,036402	15,59267	28,39117	11	29	0,975926
18	10,33683	2,25012	2,25012	15,59267	25,759246	11	29	0,975926
19	10,33683	2,38541	2,38541	15,59267	22,650692	11	29	0,975926
20	10,33683	2,68452	2,68452	15,59267	19,263184	11	29	0,975926

fig.33

Le numerose misurazioni sono state eseguite variando di volta in volta alcuni termini come la pressione e la portata dell'acqua (PLiq e QLiq), agendo sulla pompa dell'impianto, e le dimensioni dei diametri (D1 e D2) degli ugelli presenti nell'eiettore.

Il modello matematico da realizzare dovrà essere in grado di stimare in maniera più accurata possibile la portata di aria aspirata all'interno dell'eiettore (QGas), prima della miscelazione con l'acqua, che sarà quindi considerata come una variabile di uscita *y* (indicata in rosso nella *fig.33*). Tutti gli altri dati misurati come ad esempio la pressione dell'acqua (PLiq), la portata dell'acqua (QLiq) ecc. e i dati noti D1 e D2, possono essere considerati come possibili ingressi

x_i (indicati in azzurro nella *fig.33*), ognuno contraddistinto da uno specifico parametro a_i , che potranno influire in maniera più o meno significativa sulla stima della portata dell'aria aspirata.

Successivamente questi dati campionati, insieme a varie equazioni parametriche formulate saranno "forniti" a diversi algoritmi di Swarm Intelligence sviluppati su Matlab, i quali calcoleranno i valori migliori da attribuire ai diversi parametri a_i (che moltiplicano i relativi ingressi x_i) secondo diverse logiche di risoluzione basate sull' "intelligenza dello sciame" indicate nel capitolo 3, al fine di minimizzare la differenza tra l'uscita reale (misurata) e quella stimata (ricavata con l'algoritmo). L'uscita stimata deve quindi essere il più simile possibile a quella misurata, in modo tale da creare così un modello matematico che sia in grado di "prevedere" il comportamento dell'eiettore anche per altri valori di ingressi non campionati. I parametri in questo caso non assumono un'interpretazione fisica, ma sono solamente un mezzo per descrivere i vari legami (relazioni) ingresso-uscita del "sistema eiettore". Infine i risultati forniti da ciascun algoritmo verranno confrontati allo scopo di individuare l'algoritmo e il tipo di equazione parametrica che meglio rappresentano il comportamento dell'eiettore.

4.1.1 Selezione e modifica degli algoritmi di SI

In questo progetto di tesi per raggiungere l'obiettivo precedentemente esposto, ossia individuare un'equazione parametrica che rappresenti il comportamento dell'eiettore, sono stati selezionati ed utilizzati 5 algoritmi di Swarm Intelligence (tra quelli indicati nel capitolo 3), al fine di calcolare i migliori valori dei parametri da attribuire all'equazione.

Questa selezione è stata effettuata in maniera del tutto personale e arbitraria, pertanto qualora in futuro si volessero utilizzare altri tipi di algoritmi di SI, ciò risulterà possibile senza alcun tipo di vincolo. In particolare sono stati scelti i seguiti algoritmi :

- ABC – Artificial Bee Colony;
- BA – Bat Algorithm;
- FFA – FireFly Algorithm;
- GWO – Grey Wolf Optimizer;
- PSO – Particle Swarm Optimization.

I codici Matlab degli algoritmi selezionati, sono stati scaricati dal portale File Exchange di MathWorks. Di seguito si riporta per ciascun algoritmo il corrispettivo autore/sviluppatore:

ABC (S. Mostapha Kalami Heris); BA (Abhishek Gupta); FFA (S. Mostapha Kalami Heris); GWO (Seyedali Mirjalili); PSO (S. Mostapha Kalami Heris).

Successivamente i 5 algoritmi sono stati modificati, e trasformati in una *function* (funzione), al fine di "standardizzare" poi le operazioni effettuate durante le simulazioni ed utilizzare come "ingressi" gli stessi dati. Ad ogni *function* rappresentante un algoritmo sono stati forniti in ingresso i seguenti dati:

- *input* → matrice dei dati misurati, considerati come ingressi (*NxP*, dove *N* sono le misurazioni considerate e *P* corrisponde al nr. degli ingressi = nr. dei parametri);
- *output* → vettore dei dati misurati, considerati come uscite (*Nx1*, dove *N* sono le misurazioni considerate e *1* rappresenta l'unica uscita QGas);
- *dim* → valore che indica il nr. di parametri considerati ≡ *P*;
- *type_fun* → "nome" del tipo di equazione parametrica che si desidera testare;
- *Max_iter* → nr. di iterazioni massime che si desiderano far svolgere all'algoritmo*;*
- *nPop* → nr. di particelle (agenti artificiali come lupi, api, pipistrelli, ecc.) che ad ogni iterazione dell'algoritmo ricercano la posizione ottima, basandosi sulla posizione ottima trovata all'iterazione precedente: le coordinate ($c_1, c_2, \ldots, c_p$) della posizione ottima alla fine delle iterazioni, corrisponderanno ai valori da assegnare agli *P* parametri;
- *lb* → vettore contenente i valori minimi (*lower bounds*) che possono assumere ogni coordinata (*1xP*)
- *ub* → vettore contenente i valori massimi (*upper bounds*) che possono assumere ogni coordinata (*1xP*)

Per di più all'interno di ogni algoritmo è stata inserita una funzione denominata *calcola_costo* (riportata in *fig.34*).

```
function [costo,yia]=calcola_costo(posizione,input,output,type_fun)
    switch type_fun
        case 'lineare'
            yi='posizione(1)*input(:,1)+posizione(2)*input(:,2)+posizione(3)*input(:,3)';
        case 'quadratica'
            yi='posizione(1)*input(:,1).^2+posizione(2)*input(:,2).^2+posizione(3)*input(:,3).^2';
        case 'cubica'
            yi='posizione(1)*input(:,1).^3+posizione(2)*input(:,2).^3+posizione(3)*input(:,3).^3';
    end
    yia=eval(yi);
    costo=mean((output-yia).^2);
end
```

fig.34

La funzione *calcola_costo*, ad ogni iterazione dell'algoritmo di SI, riceve in ingresso la *posizione* (vettore *1xP*), che contiene i migliori valori dei parametri individuati dai vari agenti artificiali (particelle). Riceve inoltre i valori degli ingressi misurati *input* (matrice *NxP*), i valori dell'uscita misurata *output* (vettore *Nx1*) ed infine il tipo di equazione parametrica *type_fun*, che si desidera testare. Tramite la struttura *switch* e *case*, sarà possibile di volta in volta testare le varie equazioni parametriche ipotizzate.

Tale funzione è stata sviluppata per calcolare i valori dell'uscita stimata *yia* (vettore *Nx1*) corrispondenti ad ogni uscita misurata *output*, ed inoltre per calcolare l'errore quadratico medio (MSE) indicato come *costo,* che rappresenta quanto i valori dell'uscita stimata si discostano mediamente da quelli dell'uscita misurata. Questo valore di *costo*, in uscita dalla funzione *calcola_costo*, ad ogni iterazione sarà minimizzato dall'algoritmo di SI, al fine quindi di ridurre sempre di più la differenza tra le 2 uscite (misurata e stimata).

Infine ogni algoritmo (trasformato in una *function*), una volta eseguite tutte le *Max_iter*, fornirà in uscita i seguenti dati:

- *Best_Position* → vettore (*1xP*) che rappresenta i migliori valori delle coordinate (parametri) individuate dai vari agenti artificiali: valori che minimizzano le differenze tra l'uscita misurata e stimata;

- *Best_Cost* → valore di *costo* migliore (minimo) risultante;

- *Convergence_curve* → vettore (*Max_iter x 1*) che riporta tutti i valori del *costo*, calcolati ad ogni iterazione: plottando tale vettore si avrà una curva decrescente che indica appunto la minimizzazione del *costo* (MSE) durante le varie iterazioni dell'algoritmo. L'ultimo valore di tale curva corrisponderà esattamente al *Best_Cost*;

- *Output_stimata* → vettore (*Nx1*) che riporta tutti i valori dell'uscita stimata *yia*, calcolati utilizzando come variabili i valori degli ingressi misurati (vettore *input*) e come parametri i valori presenti nel vettore *Best_Position*.

4.1.2 Selezione e normalizzazione dei dati sperimentali

Una volta selezionati e modificati gli algoritmi di SI, sono stati scelti quali dati sperimentali (ottenuti con le misurazioni "sul campo") fornire agli stessi algoritmi. Come detto precedentemente i dati che potevano essere considerati come ingressi erano:

- Pressione dell'acqua (PLiq) → misurata con il sensore LN5 [bar];
- Pressione nel diffusore (PDiffusore) → misurata con il sensore LN9 [bar];
- Pressione nel serbatoio (PSerb) → misurata con il sensore LN19 [bar];
- Pressione dell'aria (PGas) → misurata con il sensore LN11 [bar];
- Portata dell'acqua (QLiq) → misurata con il sensore LN4 [m^3/h];
- D1 e D2 [mm].

In questo progetto di tesi sono stati considerati come ingressi 3 tipologie di dati:

- Pressione dell'acqua (PLiq);
- Pressione nel serbatoio (PSerb);
- Portata dell'acqua (QLiq);

mentre D1 e D2 sono stati assunti "noti". Questo vuol dire che i dati delle misurazioni forniti agli algoritmi corrispondono solamente a quelle fatte mantenendo invariati i valori dei 2 diametri degli ugelli. In particolare, sono stati considerati i dati riferiti alla configurazione degli ugelli montati attualmente sull'eiettore (D1=11mm e D2=29mm). Tali considerazioni fanno sì che la matrice degli *input* risulta avere N=64 righe (corrisponde al numero di misurazioni effettuare per il valori D1 e D2 prestabiliti) e P=3 colonne (corrisponde al numero di tipologie di dati scelti come ingressi) → *input* (*64x3*).

In particolare sono stati scelti come ingressi i dati che risultano essere più influenti, poiché la PLiq e la QLiq sono delle variabili che condizionano molto la differenza di pressione che si viene a creare all'interno dell'eiettore e quindi di conseguenza anche il valore della QGas. La PSerb rappresenta invece un vincolo che determina appunto la quantità di aria e di acqua che deve essere presente nel serbatoio stesso. Contrariamente i dati della PGas non sono stati utilizzati poiché risultavano avere valore più o meno costante per tutte le misurazioni effettuate e all'incirca pari alla pressione atmosferica (1bar).

Per quanto riguarda il vettore *output*, gli unici dati considerati come uscita, come detto precedentemente, corrispondono alle misurazioni della Portata dell'aria (QGas) effettuate dal sensore LN8 [m^3/h]→ *output* (*64x1*).

Normalizzazione

Dopo aver individuato la matrice degli *input* e il vettore dell'*output*, i dati sperimentali contenuti in essi sono stati normalizzati. Tramite la tecnica della **normalizzazione**, definita anche **Min-Max Scaling**, tutti i dati sono stati ridimensionati su un intervallo di valori fisso [0,1], al fine di essere poi confrontati (una volta forniti agli algoritmi) in una maniera più corretta. La normalizzazione rendendo i dati adimensionali, consente infatti una migliore comparazione degli stessi, soprattutto quando presentano diverse unità di misura. Ridimensionando tutti i dati in un unico intervallo di valori prestabilito, inoltre, ciascuna tipologia di dati di ingresso, a parità del valore dei parametri moltiplicativi, risulterà avere la stessa incidenza sul calcolo dell'uscita stimata.

Considerando ad esempio la matrice degli *input* formata da 3 vettori *input_1*, *input_2* e *input_3*, rappresentanti le misurazioni della PLiq, della PSerb e della QLiq, un generico valore della PSerb, viene normalizzato nel seguente modo:

$$z_{2i} = \frac{x_{2i} - x_{2min}}{x_{2max} - x_{2min}}$$

dove x_{2i} rappresenta un valore generico non normalizzato (grezzo) del vettore *input_2* , mentre x_{2min} e x_{2max} sono i valori minimi e massimi misurati, sempre contenuti nel vettore *input_2*. Con il termine **Scaling** precedentemente citato si indica infatti l'aggiungere/sottrarre una costante e

poi moltiplicare/dividere il risultato per un'altra costante, in modo che tutti i dati possano trovarsi tra determinati valori minimi e massimi.

Al fine di non rendere nessun valore normalizzato uguale al valore minimo (0) e al valore massimo (1) del range precedentemente considerato, la formula effettivamente utilizzata per la normalizzazione dei dati risulta essere la seguente:

$$z_i = \frac{x_i - (x_{min} * 0{,}9)}{(x_{max} * 1{,}1) - (x_{min} * 0{,}9)}$$

4.1.3 Simulazione e valutazione dei modelli parametrici

Oltre ad aver distinto sia i vari algoritmi di SI, sia i dati sperimentali da utilizzare per effettuare l'identificazione parametrica, allo scopo di procedere con le simulazioni, sono stati definiti anche altri dati precedentemente riportati, quali:

- *dim* ≡ *P* = 3 → poiché sono state considerate 3 tipologie di ingresso (PLiq, PSerb e QLiq) si è deciso di utilizzare 3 parametri moltiplicativi nelle equazioni parametriche;

- *type_fun* → si è deciso di testare 3 diverse equazioni parametriche (al fine poi di individuare la migliore) denominate e caratterizzate nel seguente modo:

  1. LINEARE

     $a(1) * input(:,1) + a(2) * input(:,2) + a(3) * input(:,3)$

  2. QUADRATICA

     $a(1) * input(:,1).\hat{}2 + a(2) * input(:,2).\hat{}2 + a(3) * input(:,3).\hat{}2$

  3. CUBICA

     $a(1) * input(:,1).\hat{}3 + a(2) * input(:,2).\hat{}3 + a(3) * input(:,3).\hat{}3$

  Dove *a(1)*, *a(2)* e *a(3)* rappresentano i 3 parametri il cui valore corrisponderà alle "coordinate" della migliore posizione trovata dagli agenti artificiali durante l'esecuzione dell'algoritmo;

- *lb* → i valori minimi che possono assumere le 3 coordinate/parametri, sono stati tutti fissati pari a -1: [-1,-1,-1] ;

- *ub* → i valori massimi che possono assumere le 3 coordinate/parametri, sono stati tutti fissati pari a 1: [1,1,1] ;
- *Max_iter* → sono stati assunti diversi valori del nr. di iterazioni: 25, 50, 75 e 100 ;
- *nPop* → così come per le iterazioni, anche per il nr. di agenti artificiali (particelle) sono stati assunti diversi valori: 25, 50, 75 e 100.

In particolare, partendo dalla scelta del tipo di equazione parametrica che si desiderava testare, di volta in volta, ad ogni prova/simulazione, sono stati fatti variare sia il numero di iterazioni, sia il numero di particelle. Alla fine dell'esecuzione di ciascuno dei 5 algoritmi scelti, sono state plottate le 5 *output_stimate* in relazione all'*output* (misurata) e le 5 *Convergence_curve* che mostravano appunto la velocità di ciascun algoritmo di convergere verso un valore minimo di MSE. Tutto ciò al fine di visualizzare ed individuare, in primo luogo, quale dei 5 algoritmi risultava essere più adeguato al caso in esame, effettuando quindi un'analisi qualitativa.

A titolo informativo, nella sezione "**Appendice A**" è riporto il codice dell'algoritmo PSO. Come detto in precedenza gli algoritmi selezionati essendo stati "standardizzati" e "uniformati", risultano essere simili, per quanto riguarda i dati forniti in ingresso e i dati in uscita, la differenza principale sta nel modo in cui essi operano, ossia nel procedimento con cui "guidano" gli agenti artificiali (particelle) verso la posizione ottima del problema preso in esame (quella che minimizza appunto l'MSE).

Indici di valutazione

Ad ogni simulazione effettuata, per individuare in maniera quantitativa l'algoritmo di modellazione parametrica migliore, tra i 5 proposti, sono stati adoperati 2 indici di valutazione della bontà dei modelli: indice AIC (Akaike Information Criterion) e BIC (Bayesian Information Criterion). Tali indici sono stati utilizzati appunto per la comparazione dei modelli.

Questi ultimi sono definiti dalle seguenti formule:

$$AIC = N * ln\left(\frac{RSS}{N}\right) + 2P \qquad BIC = N * ln\left(\frac{RSS}{N}\right) + P * ln(N)$$

dove *N* è il numero di osservazioni (misurazioni) considerate, *P* è il numero di parametri inclusi nel modello (equazione parametrica) e *RSS* rappresenta la sommatoria degli errori quadratici,

intesi come differenza tra le misurazioni dell'uscita reale e i corrispondenti valori dell'uscita stimata, elevata al quadrato:

$$RSS = \sum_{i=1}^{N} e_i^2$$

Un modello risulta essere migliore rispetto gli altri con cui è stato confrontato, quando questo ultimo presenta un valore di AIC o BIC minore. Due modelli possono definirsi differenti quando tra di loro sussiste una differenza degli indici (AIC o BIC) almeno pari a 2 ($\Delta \geq 2$) (Raftery 1995). Se tale differenza è presente, il modello con AIC o BIC più basso sarà da preferire.

Sia l'AIC che il BIC sono calcolati a partire dal logaritmo naturale di *RSS* diviso per il numero di osservazioni (misurazioni) *N*. Tale valore viene poi incrementato di un secondo termine in funzione di *P*, che penalizza quindi il modello. I due indici differiscono in questa funzione di penalizzazione.

Qualora la dimensione del campione di osservazioni fosse troppo piccola rispetto al numero di parametri considerati ($N/P < 40$), per entrambi gli indici si applicano le seguenti correzioni (Burnham e Anderson 2004):

$$AICc = AIC + \frac{2P(P+1)}{N-P-1} \qquad BICc = BIC + \frac{P*ln(N)*(P+1)}{N-P-1}$$

dove AICc e BICc corrispondono ai 2 indici corretti. Nel caso in esame, essendo verificata la suddetta situazione, (poiché $N=64$ e $P=3$ → $N/P \cong 21 < 40$), sono state calcolate entrambe le "tipologie" di indici, cioè con e senza correzione.

4.2 Equazione parametrica LINEARE

Le simulazioni sono state eseguite a partire dall'equazione parametrica **lineare**:

$$a(1) * input(:,1) + a(2) * input(:,2) + a(3) * input(:,3)$$

Ogni simulazione è stata svolta facendo variare prima il numero di agenti artificiali (particelle *nPop*) e poi in numero di iterazioni (*Max_iter*). In *fig.35* e *fig.36* sono riportate le uscite stimate da ciascuno dei 5 algoritmi scelti e l'uscita reale (campionata nelle 64 misurazioni), considerando nel primo caso 50 iterazioni – 25 particelle (*fig.35*) e nel secondo caso 100 iterazioni – 50 particelle (*fig.36*).

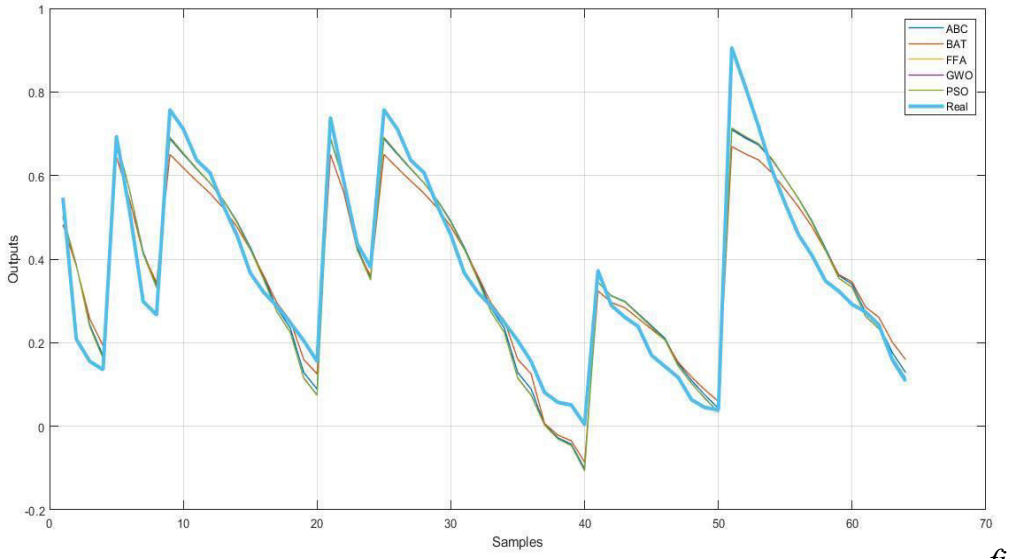


fig.35

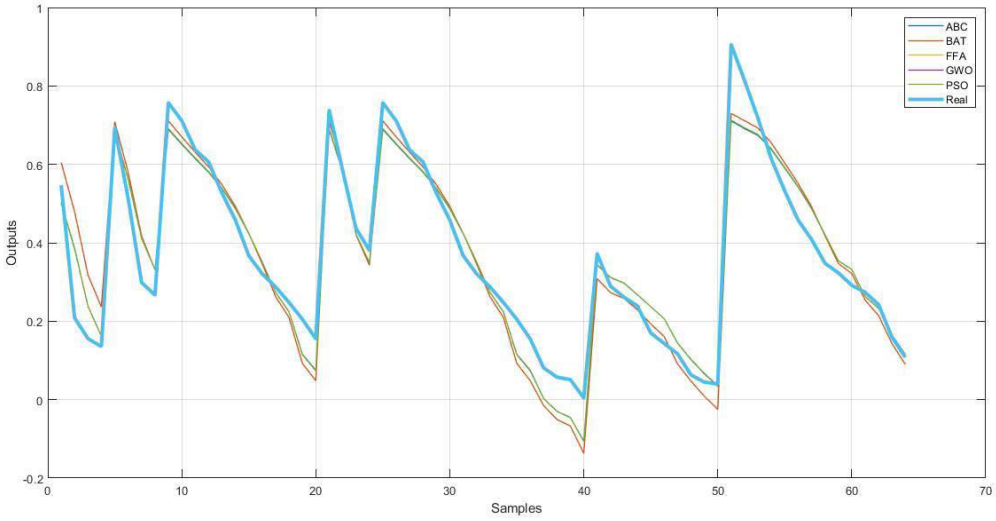


fig.36

Confrontando in entrambi i casi l'*output* misurata (Real) con le *output_stimate* dai 5 algoritmi (ABC, BAT, FFA, GWO, PSO), si può notare che queste ultime, seppur differendo in maniera più marcata in determinati punti, presentano un andamento simile all'uscita reale. Tale affermazione sta ad indicare che il legame ipotizzato (**lineare**), può rappresentare in maniera abbastanza adeguata il comportamento dell'eiettore. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

Nelle tabelle sottostanti (*tab.3.1* e *tab.3.2*) sono riportati i valori dei vari indici di valutazione, calcolati per ogni simulazione effettuata.

SIMULAZ	nr. ITER	nr. PART	ABC				BAT				FFA			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-331,49	-331,09	-325,01	-324,18	-322,19	-321,79	-315,72	-314,89	-348,16	-347,76	-341,68	-340,85
2	25	50	-348,05	-347,65	-341,57	-340,74	-321,23	-320,83	-314,75	-313,92	-348,16	-347,76	-341,68	-340,85
3	25	75	-348,09	-347,69	-341,61	-340,78	-336,19	-335,79	-329,71	-328,88	-348,16	-347,76	-341,68	-340,85
4	25	100	-348,11	-347,71	-341,63	-340,80	-334,91	-334,51	-328,43	-327,60	-348,16	-347,76	-341,68	-340,85
5	50	25	-347,47	-347,07	-341,00	-340,16	-336,36	-335,96	-329,89	-329,06	-348,16	-347,76	-341,68	-340,85
6	50	50	-348,10	-347,70	-341,62	-340,79	-276,73	-276,33	-270,25	-269,42	-348,16	-347,76	-341,68	-340,85
7	50	75	-348,15	-347,75	-341,68	-340,85	-293,04	-292,64	-286,56	-285,73	-348,16	-347,76	-341,68	-340,85
8	50	100	-348,16	-347,76	-341,68	-340,85	-335,92	-335,52	-329,44	-328,61	-348,16	-347,76	-341,68	-340,85
9	75	25	-348,06	-347,66	-341,58	-340,75	-338,36	-337,96	-331,88	-331,05	-348,16	-347,76	-341,68	-340,85
10	75	50	-348,14	-347,74	-341,66	-340,83	-318,14	-317,74	-311,67	-310,83	-348,16	-347,76	-341,68	-340,85
11	75	75	-348,16	-347,76	-341,68	-340,85	-332,38	-331,98	-325,91	-325,07	-348,16	-347,76	-341,68	-340,85
12	75	100	-348,16	-347,76	-341,68	-340,85	-335,11	-334,71	-328,63	-327,80	-348,16	-347,76	-341,68	-340,85
13	100	25	-347,94	-347,54	-341,46	-340,63	-286,92	-286,52	-280,45	-279,62	-348,16	-347,76	-341,68	-340,85
14	100	50	-348,14	-347,74	-341,66	-340,83	-327,47	-327,07	-321,00	-320,17	-348,16	-347,76	-341,68	-340,85
15	100	75	-348,16	-347,76	-341,68	-340,85	-294,86	-294,46	-288,39	-287,56	-348,16	-347,76	-341,68	-340,85
16	100	100	-348,16	-347,76	-341,68	-340,85	-330,27	-329,87	-323,80	-322,97	-348,16	-347,76	-341,68	-340,85

tab.3.1

SIMULAZ	nr. ITER	nr. PART	GWO				PSO			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-343,87	-343,47	-337,39	-336,56	-347,38	-346,98	-340,91	-340,08
2	25	50	-348,05	-347,65	-341,57	-340,74	-348,05	-347,65	-341,57	-340,74
3	25	75	-348,16	-347,76	-341,68	-340,85	-348,11	-347,71	-341,63	-340,80
4	25	100	-348,15	-347,75	-341,68	-340,85	-348,14	-347,74	-341,67	-340,83
5	50	25	-348,15	-347,75	-341,68	-340,85	-348,15	-347,75	-341,68	-340,85
6	50	50	-343,56	-343,16	-337,09	-336,25	-348,16	-347,76	-341,68	-340,85
7	50	75	-348,15	-347,75	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
8	50	100	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
9	75	25	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
10	75	50	-343,71	-343,31	-337,23	-336,40	-348,16	-347,76	-341,68	-340,85
11	75	75	-343,98	-343,58	-337,50	-336,67	-348,16	-347,76	-341,68	-340,85
12	75	100	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
13	100	25	-342,60	-342,20	-336,13	-335,30	-348,16	-347,76	-341,68	-340,85
14	100	50	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
15	100	75	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85
16	100	100	-348,16	-347,76	-341,68	-340,85	-348,16	-347,76	-341,68	-340,85

tab.3.2

Come si può osservare, confrontando i relativi valori dei vari indici di ciascun algoritmo, questi risultano essere pressoché identici, a parte quelli del BAT. Ciò significa che l'ABC, il FFA, il GWO e il PSO creano dei modelli parametrici molto simili tra loro, che risultano essere migliori del modello generato dal BAT, poiché presentano valori di indici più bassi.

Nelle precedenti tabelle (*tab.3.1* e *tab.3.2*) sono stati riportati i valori degli indici considerati, approssimati alla seconda cifra decimale. Se si considerassero più cifre decimali, sarebbe più semplice individuare quale dei 5 algoritmi genera il modello migliore (con valori di indici più bassi). Di seguito, nella *tab.4* sono riportati i migliori modelli generati dai relativi algoritmi ad ogni simulazione effettuata, con annessi i valori ottimi dei parametri *a(1)*, *a(2)* e *a(3)*.

SIMULAZ	nr. ITER	nr. PART	ALGORITMO migliore	*a(1)*	*a(2)*	*a(3)*
1	25	25	FFA	-0,1099	-0,7753	1,0000
2	25	50	FFA	-0,1096	-0,7760	1,0000
3	25	75	FFA	-0,1098	-0,7754	1,0000
4	25	100	FFA	-0,1097	-0,7756	1,0000
5	50	25	FFA	-0,1098	-0,7755	1,0000
6	50	50	FFA	-0,1096	-0,7757	1,0000
7	50	75	FFA	-0,1097	-0,7755	1,0000
8	50	100	FFA	-0,1097	-0,7755	1,0000
9	75	25	PSO	-0,1097	-0,7756	1,0000
10	75	50	PSO	-0,1097	-0,7756	1,0000
11	75	75	FFA	-0,1097	-0,7756	1,0000
12	75	100	PSO	-0,1097	-0,7756	1,0000
13	100	25	PSO	-0,1097	-0,7756	1,0000
14	100	50	PSO	-0,1097	-0,7756	1,0000
15	100	75	PSO	-0,1097	-0,7756	1,0000
16	100	100	PSO	-0,1097	-0,7756	1,0000

tab.4

Dalla *tab.4* si evince che al variare del numero di particelle e di iterazioni, i valori ottimi da attribuire ai parametri rimangono praticamente gli stessi. Per quanto riguarda invece la tipologia migliore di algoritmo (seppur i due forniscono risultati molto simili come detto precedentemente), aumentando il numero di iterazioni sarebbe più opportuno utilizzare il modello fornito dall'algoritmo PSO, piuttosto che quello fornito dal FFA.

Effettuando un'analisi più qualitativa, andando quindi ad esaminare le *Convergence_curve* di ciascun algoritmo, si possono confrontare le diverse velocità di convergenza ed i valori minimi di MSE che gli stessi riescono a garantire. In *fig.37* e *fig.38* sono riportate le *Convergence_curve* risultanti dalle simulazioni effettuate, considerando nel primo caso 50 iterazioni – 25 particelle e nel secondo caso 100 iterazioni – 50 particelle.

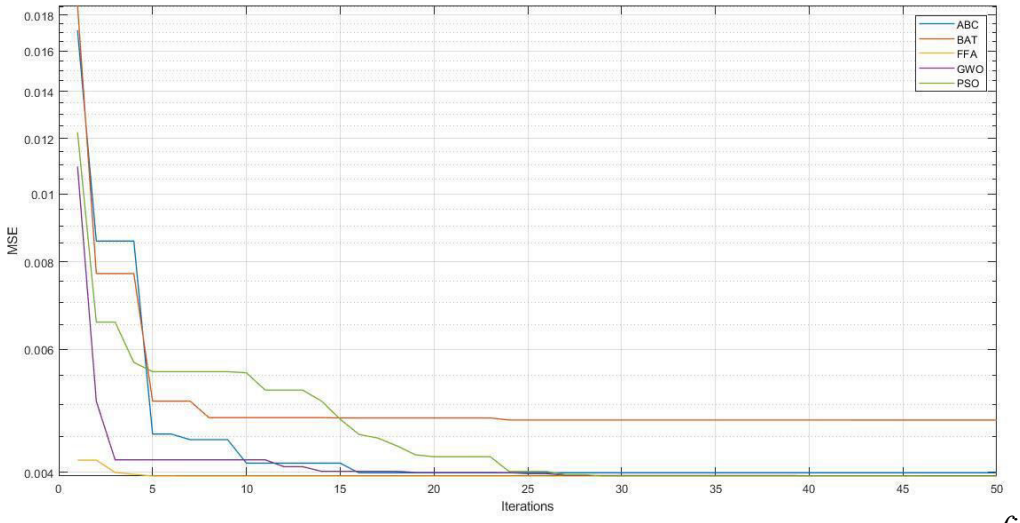


fig.37

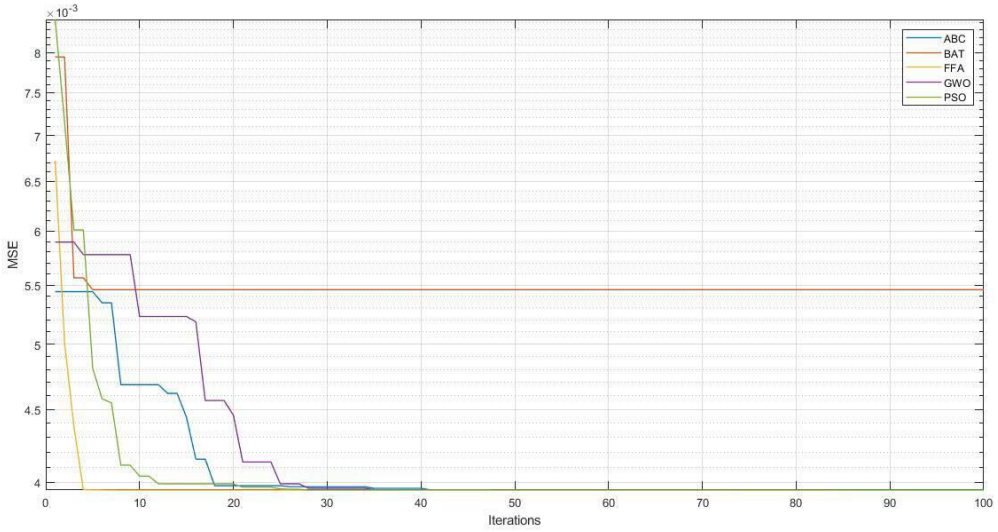


fig.38

In entrambi i casi si può osservare che il FFA risulta avere una velocità di convergenza migliore (più elevata) verso il valore minimo di MSE, rispetto agli altri 4 algoritmi. Il BAT, pur presentando una convergenza abbastanza elevata (entro le prime 10 iterazioni), converge ad un valore di MSE maggiore e risulta pertanto essere il peggiore come indicato anche in precedenza. All'incirca l'ABC, il FFA, il GWO e il PSO convergono ad un MSE pari a 0,004. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

4.3 Equazione parametrica QUADRATICA

Dopo aver effettuato tutte le simulazioni considerando l'equazione parametrica **lineare**, sono state eseguite altre simulazioni valutando l'equazione parametrica **quadratica**:

$$a(1) * input(:,1).\hat{}2 + a(2) * input(:,2).\hat{}2 + a(3) * input(:,3).\hat{}2$$

Anche in questo caso, ogni simulazione è stata svolta facendo variare prima il numero di agenti artificiali (particelle *nPop*) e poi in numero di iterazioni (*Max_iter*). In *fig.39* e *fig.40* sono riportate le uscite stimate da ciascuno dei 5 algoritmi scelti e l'uscita reale (campionata nelle 64 misurazioni), considerando nel primo caso 50 iterazioni – 75 particelle (*fig.39*) e nel secondo caso 100 iterazioni – 25 particelle (*fig.40*).

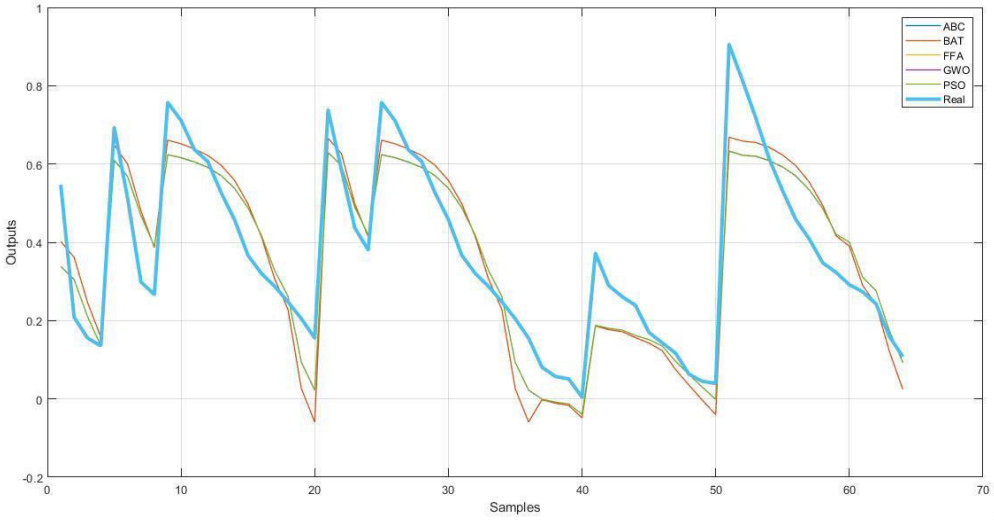


fig.39

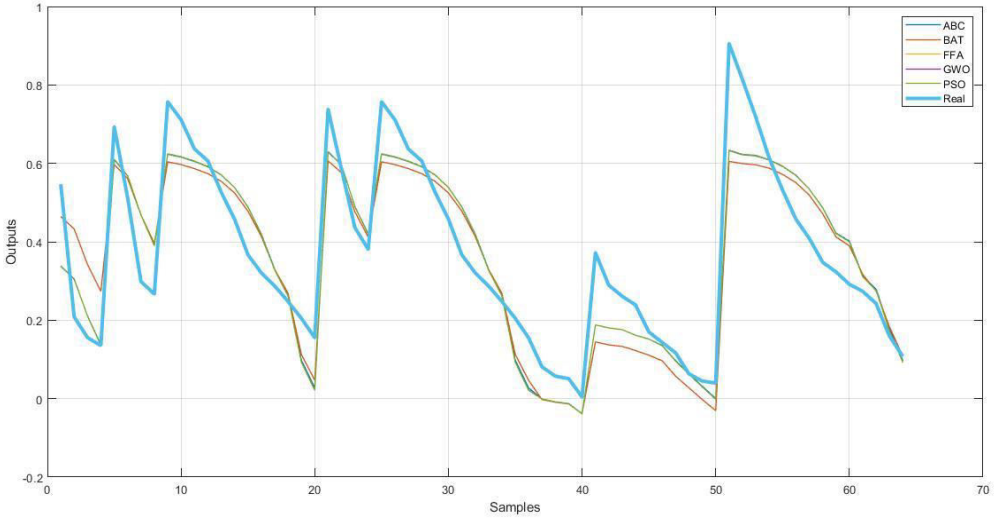


fig.40

Confrontando in entrambi i casi l'*output* misurata (Real) con le *output_stimate* dai 5 algoritmi (ABC, BAT, FFA, GWO, PSO), si può notare che queste ultime differiscono in maniera maggiore, rispetto alle curve ottenute considerando l'equazione parametrica **lineare** (*fig.35* e *fig.36).* Tale affermazione sta ad indicare che il legame ipotizzato (**quadratico**), rappresenta in maniera peggiore il comportamento dell'eiettore rispetto al legame **lineare**. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

Nelle tabelle sottostanti (*tab5.1* e *tab5.2*) sono riportati i valori dei vari indici di valutazione, calcolati per ogni simulazione effettuata.

SIMULAZ	nr. ITER	nr. PART	ABC				BAT				FFA			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-294,24	-293,84	-287,76	-286,93	-291,37	-290,97	-284,89	-284,06	-294,42	-294,02	-287,95	-287,12
2	25	50	-294,25	-293,85	-287,78	-286,94	-285,84	-285,44	-279,37	-278,53	-294,42	-294,02	-287,95	-287,12
3	25	75	-294,35	-293,95	-287,87	-287,04	-283,12	-282,72	-276,64	-275,81	-294,42	-294,02	-287,95	-287,12
4	25	100	-294,34	-293,94	-287,86	-287,03	-274,73	-274,33	-268,25	-267,42	-294,42	-294,02	-287,95	-287,12
5	50	25	-294,38	-293,98	-287,91	-287,08	-286,05	-285,65	-279,57	-278,74	-294,42	-294,02	-287,95	-287,12
6	50	50	-294,42	-294,02	-287,95	-287,11	-289,24	-288,84	-282,76	-281,93	-294,42	-294,02	-287,95	-287,12
7	50	75	-294,42	-294,02	-287,95	-287,12	-286,43	-286,03	-279,96	-279,12	-294,42	-294,02	-287,95	-287,12
8	50	100	-294,42	-294,02	-287,95	-287,12	-283,05	-282,65	-276,58	-275,75	-294,42	-294,02	-287,95	-287,12
9	75	25	-294,40	-294,00	-287,92	-287,09	-281,41	-281,01	-274,94	-274,11	-294,42	-294,02	-287,95	-287,12
10	75	50	-294,42	-294,02	-287,95	-287,12	-292,26	-291,86	-285,78	-284,95	-294,42	-294,02	-287,95	-287,12
11	75	75	-294,42	-294,02	-287,95	-287,12	-279,13	-278,73	-272,66	-271,83	-294,42	-294,02	-287,95	-287,12
12	75	100	-294,42	-294,02	-287,95	-287,12	-289,89	-289,49	-283,41	-282,58	-294,42	-294,02	-287,95	-287,12
13	100	25	-294,40	-294,00	-287,92	-287,09	-281,53	-281,13	-275,06	-274,22	-294,42	-294,02	-287,95	-287,12
14	100	50	-294,42	-294,02	-287,95	-287,12	-274,52	-274,12	-268,04	-267,21	-294,42	-294,02	-287,95	-287,12
15	100	75	-294,42	-294,02	-287,95	-287,12	-286,98	-286,58	-280,50	-279,67	-294,42	-294,02	-287,95	-287,12
16	100	100	-294,42	-294,02	-287,95	-287,12	-275,38	-274,98	-268,90	-268,07	-294,42	-294,02	-287,95	-287,12

tab.5.1

SIMULAZ	nr. ITER	nr. PART	GWO				PSO			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-294,42	-294,02	-287,94	-287,11	-294,39	-293,99	-287,91	-287,08
2	25	50	-292,94	-292,54	-286,46	-285,63	-294,42	-294,02	-287,94	-287,11
3	25	75	-291,33	-290,93	-284,85	-284,02	-294,38	-293,98	-287,90	-287,07
4	25	100	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,94	-287,11
5	50	25	-294,42	-294,02	-287,95	-287,11	-294,42	-294,02	-287,95	-287,12
6	50	50	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
7	50	75	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
8	50	100	-294,42	-294,02	-287,95	-287,11	-294,42	-294,02	-287,95	-287,12
9	75	25	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
10	75	50	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
11	75	75	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
12	75	100	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
13	100	25	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
14	100	50	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
15	100	75	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12
16	100	100	-294,42	-294,02	-287,95	-287,12	-294,42	-294,02	-287,95	-287,12

tab.5.2

Come si può osservare, confrontando i relativi valori dei vari indici di ciascun algoritmo, questi risultano essere pressoché identici, a parte quelli del BAT. Ciò significa che anche in questo caso, l'ABC, il FFA, il GWO e il PSO creano dei modelli parametrici molto simili tra loro, che risultano essere migliori del modello generato dal BAT, poiché presentano valori di indici più bassi. Effettuando un'ulteriore analisi, si può notare che tali valori risultano essere maggiori di quelli calcolati considerando il legame **lineare** (*tab.3.1* e *tab.3.2*) anziché **quadratico**. Ciò riconferma il fatto che il legame **quadratico** fornisce risultati peggiori del legame **lineare**.

Nelle precedenti tabelle (*tab.5.1* e *tab.5.2*) sono stati riportati i valori degli indici considerati, approssimati alla seconda cifra decimale. Se si considerassero più cifre decimali, sarebbe più semplice individuare quale dei 5 algoritmi genera il modello migliore (con valori di indici più bassi). Di seguito, nella *tab.6* sono riportati i migliori modelli generati dai relativi algoritmi ad ogni simulazione effettuata, con annessi i valori ottimi dei parametri *a(1)*, *a(2)* e *a(3)*.

SIMULAZ	nr. ITER	nr. PART	ALGORITMO migliore	*a(1)*	*a(2)*	*a(3)*
1	25	25	FFA	-0,1086	-0,8038	1,0000
2	25	50	FFA	-0,1085	-0,8042	1,0000
3	25	75	FFA	-0,1085	-0,8041	1,0000
4	25	100	FFA	-0,1085	-0,8041	1,0000
5	50	25	FFA	-0,1085	-0,8041	1,0000
6	50	50	PSO	-0,1085	-0,8040	1,0000
7	50	75	FFA	-0,1085	-0,8041	1,0000
8	50	100	PSO	-0,1085	-0,8041	1,0000
9	75	25	PSO	-0,1085	-0,8041	1,0000
10	75	50	PSO	-0,1085	-0,8040	1,0000
11	75	75	PSO	-0,1085	-0,8040	1,0000
12	75	100	PSO	-0,1085	-0,8040	1,0000
13	100	25	PSO	-0,1085	-0,8040	1,0000
14	100	50	PSO	-0,1085	-0,8040	1,0000
15	100	75	PSO	-0,1085	-0,8040	1,0000
16	100	100	PSO	-0,1085	-0,8040	1,0000

tab.6

Dalla *tab.6* si evince che al variare del numero di particelle e di iterazioni, i valori ottimi da attribuire ai parametri rimangono praticamente gli stessi. Per quanto riguarda invece la tipologia migliore di algoritmo (seppur i due forniscono risultati molto simili come detto precedentemente), aumentando il numero di iterazioni sarebbe più opportuno utilizzare il modello fornito dall'algoritmo PSO, piuttosto che quello fornito dal FFA.

Effettuando un'analisi più qualitativa, andando quindi ad esaminare le *Convergence_curve* di ciascun algoritmo, si possono confrontare le diverse velocità di convergenza ed i valori minimi di MSE che gli stessi riescono a garantire. In *fig.41* e *fig.42* sono riportate le *Convergence_curve* risultanti dalle simulazioni effettuate, considerando nel primo caso 50 iterazioni – 75 particelle e nel secondo caso 100 iterazioni – 25 particelle.

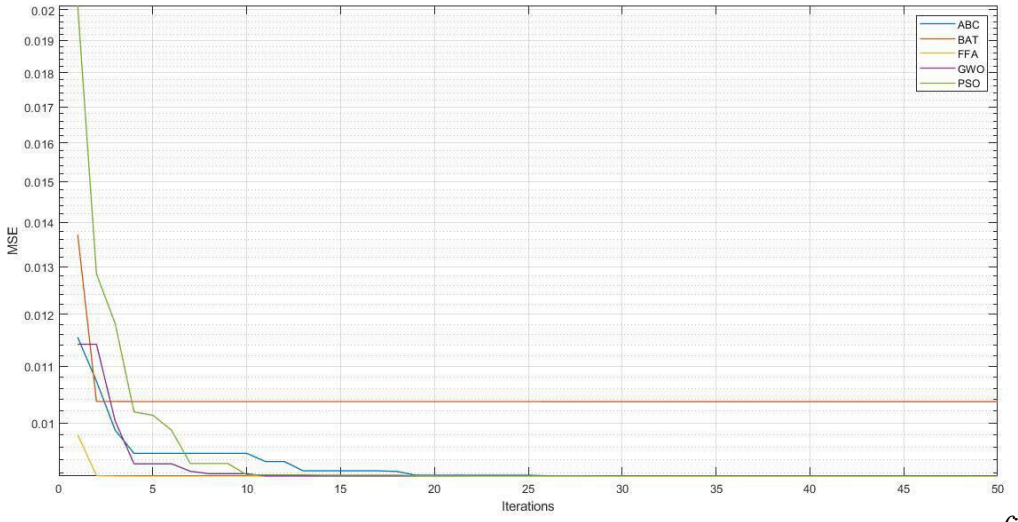


fig.41

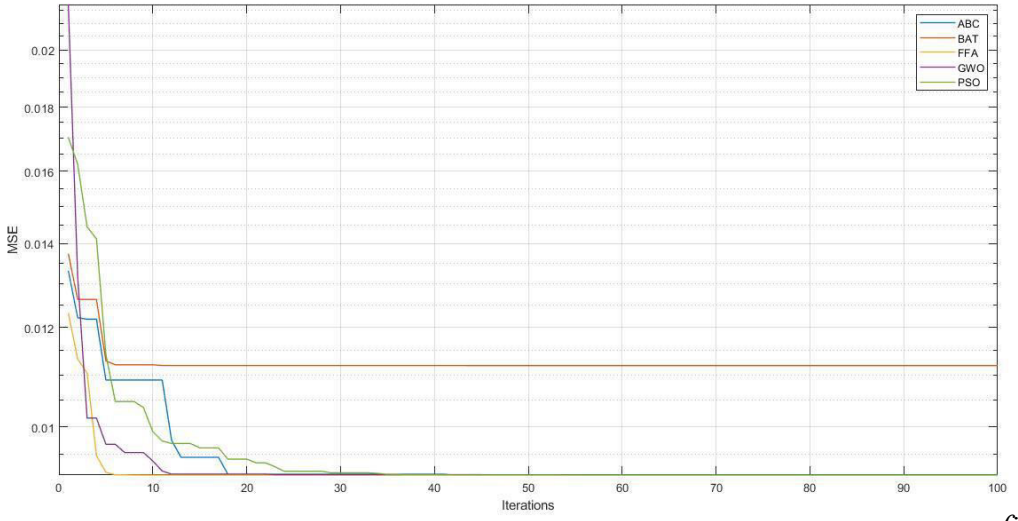


fig.42

In entrambi i casi si può osservare che il FFA risulta avere una velocità di convergenza migliore (più elevata) verso il valore minimo di MSE, rispetto agli altri 4 algoritmi. Il BAT, pur presentando una convergenza abbastanza elevata (entro le prime 10 iterazioni), converge ad un valore di MSE maggiore e risulta pertanto essere il peggiore come indicato anche in precedenza. All'incirca l'ABC, il FFA, il GWO e il PSO convergono ad un MSE pari a 0,009. Tale valore risulta superiore all'MSE minimo riscontrato considerando l'equazione parametrica **lineare** (0,004) e fornisce quindi un'ulteriore conferma della migliore bontà del legame **lineare**. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

4.4 Equazione parametrica CUBICA

Dopo aver effettuato tutte le simulazioni considerando l'equazione parametrica **quadratica**, sono state eseguite altre simulazioni valutando l'ultimo legame ipotizzato: equazione parametrica **cubica**:

$$a(1) * input(:,1).\hat{}3 + a(2) * input(:,2).\hat{}3 + a(3) * input(:,3).\hat{}3$$

Anche in questo ultimo caso, ogni simulazione è stata svolta facendo variare prima il numero di agenti artificiali (particelle *nPop*) e poi in numero di iterazioni (*Max_iter*). In *fig.43* e *fig.44* sono riportate le uscite stimate da ciascuno dei 5 algoritmi scelti e l'uscita reale (campionata nelle 64 misurazioni), considerando nel primo caso 25 iterazioni – 75 particelle (*fig.43*) e nel secondo caso 75 iterazioni – 50 particelle (*fig.44*).

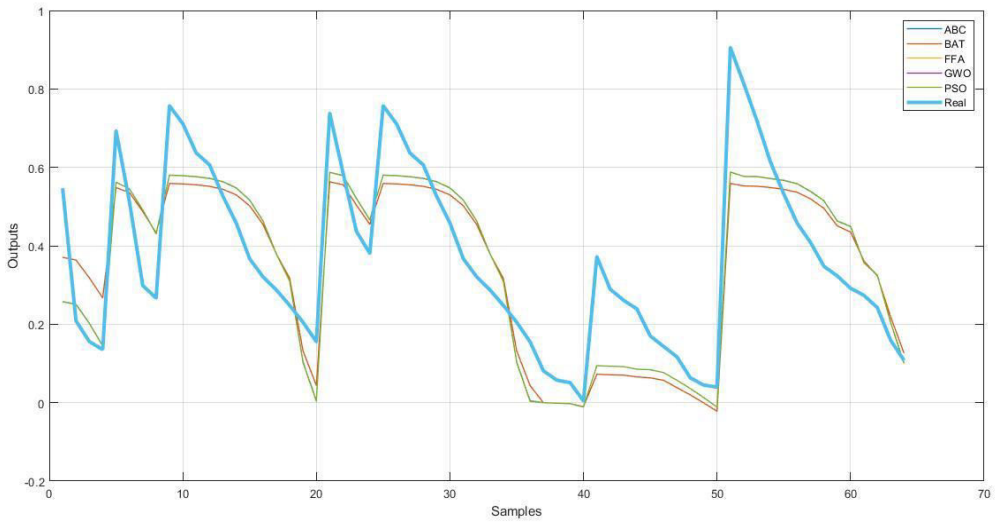


fig.43

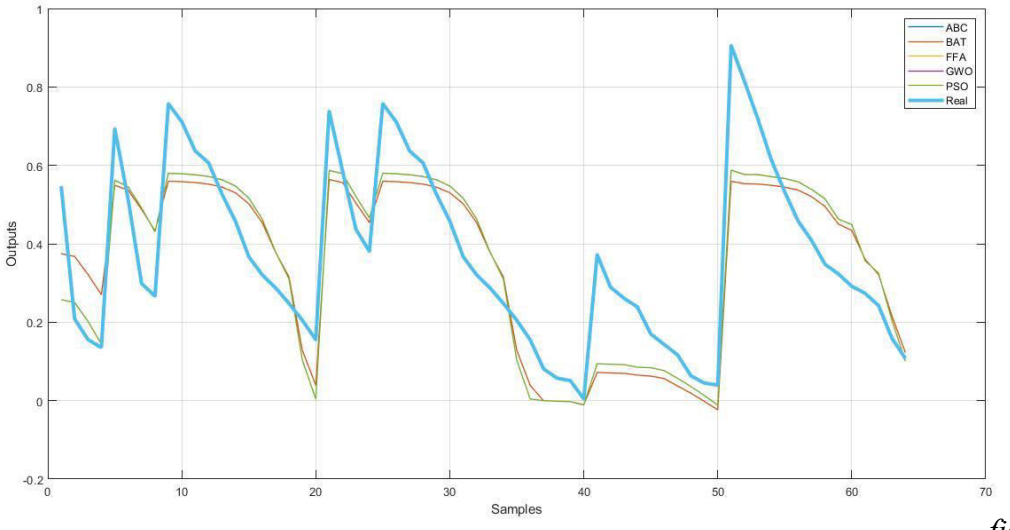


fig.44

Confrontando in entrambi i casi l'*output* misurata (Real) con le *output_stimate* dai 5 algoritmi (ABC, BAT, FFA, GWO, PSO), si può notare che queste ultime differiscono in maniera maggiore, rispetto alle curve ottenute considerando le equazioni parametriche precedenti: **lineare** (*fig.35* e *fig.36*) e **quadratica** (*fig.39* e *fig.40*). Tale affermazione sta ad indicare che il legame ipotizzato (**cubico**), rappresenta in maniera peggiore il comportamento dell'eiettore, sia rispetto al legame **lineare**, che al legame **quadratico**. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

Nelle tabelle sottostanti (*tab7.1* e *tab7.2*) sono riportati i valori dei vari indici di valutazione, calcolati per ogni simulazione effettuata.

SIMULAZ	nr. ITER	nr. PART	ABC				BAT				FFA			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-259,64	-259,24	-253,16	-252,33	-256,53	-256,13	-250,05	-249,22	-259,81	-259,41	-253,33	-252,50
2	25	50	-259,80	-259,40	-253,33	-252,49	-256,67	-256,27	-250,19	-249,36	-259,81	-259,41	-253,33	-252,50
3	25	75	-259,81	-259,41	-253,33	-252,50	-255,21	-254,81	-248,74	-247,90	-259,81	-259,41	-253,33	-252,50
4	25	100	-259,80	-259,40	-253,33	-252,49	-239,99	-239,59	-233,51	-232,68	-259,81	-259,41	-253,33	-252,50
5	50	25	-259,75	-259,35	-253,27	-252,44	-259,70	-259,30	-253,22	-252,39	-259,81	-259,41	-253,33	-252,50
6	50	50	-259,81	-259,41	-253,33	-252,50	-255,93	-255,53	-249,46	-248,62	-259,81	-259,41	-253,33	-252,50
7	50	75	-259,81	-259,41	-253,33	-252,50	-254,03	-253,63	-247,55	-246,72	-259,81	-259,41	-253,33	-252,50
8	50	100	-259,81	-259,41	-253,33	-252,50	-259,06	-258,66	-252,59	-251,75	-259,81	-259,41	-253,33	-252,50
9	75	25	-259,80	-259,40	-253,32	-252,49	-230,98	-230,58	-224,50	-223,67	-259,81	-259,41	-253,33	-252,50
10	75	50	-259,81	-259,41	-253,33	-252,50	-255,09	-254,69	-248,62	-247,79	-259,81	-259,41	-253,33	-252,50
11	75	75	-259,81	-259,41	-253,33	-252,50	-258,99	-258,59	-252,51	-251,68	-259,81	-259,41	-253,33	-252,50
12	75	100	-259,81	-259,41	-253,33	-252,50	-256,61	-256,21	-250,13	-249,30	-259,81	-259,41	-253,33	-252,50
13	100	25	-259,80	-259,40	-253,33	-252,50	-250,03	-249,63	-243,55	-242,72	-259,81	-259,41	-253,33	-252,50
14	100	50	-259,81	-259,41	-253,33	-252,50	-249,52	-249,12	-243,04	-242,21	-259,81	-259,41	-253,33	-252,50
15	100	75	-259,81	-259,41	-253,33	-252,50	-244,61	-244,21	-238,14	-237,30	-259,81	-259,41	-253,33	-252,50
16	100	100	-259,81	-259,41	-253,33	-252,50	-258,01	-257,61	-251,53	-250,70	-259,81	-259,41	-253,33	-252,50

tab.7.1

SIMULAZ	nr. ITER	nr. PART	GWO				PSO			
			AIC	*AICc*	*BIC*	*BICc*	*AIC*	*AICc*	*BIC*	*BICc*
1	25	25	-259,11	-258,71	-252,63	-251,80	-259,70	-259,30	-253,22	-252,39
2	25	50	-259,81	-259,41	-253,33	-252,50	-259,79	-259,39	-253,32	-252,49
3	25	75	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
4	25	100	-259,77	-259,37	-253,29	-252,46	-259,80	-259,40	-253,33	-252,50
5	50	25	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
6	50	50	-259,68	-259,28	-253,20	-252,37	-259,81	-259,41	-253,33	-252,50
7	50	75	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
8	50	100	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
9	75	25	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
10	75	50	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
11	75	75	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
12	75	100	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
13	100	25	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
14	100	50	-259,77	-259,37	-253,29	-252,46	-259,81	-259,41	-253,33	-252,50
15	100	75	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50
16	100	100	-259,81	-259,41	-253,33	-252,50	-259,81	-259,41	-253,33	-252,50

tab.7.2

Come si può osservare, confrontando i relativi valori dei vari indici di ciascun algoritmo, questi risultano essere pressoché identici, a parte quelli del BAT. Ciò significa che anche in questo caso, l'ABC, il FFA, il GWO e il PSO creano dei modelli parametrici molto simili tra loro, che risultano essere migliori del modello generato dal BAT, poiché presentano valori di indici più bassi. Effettuando un'ulteriore analisi si può notare che tali valori risultano essere maggiori di quelli calcolati considerando il legame **lineare** (*tab.3.1* e *tab.3.2*) e/o **quadratico** (*tab.5.1* e *tab.5.2*) anziché **cubico**. Ciò riconferma il fatto che il legame **cubico** fornisce risultati peggiori sia del legame **lineare**, che del legame **quadratico**.

Nelle precedenti tabelle (*tab.7.1* e *tab.7.2*) sono stati riportati i valori degli indici considerati, approssimati alla seconda cifra decimale. Se si considerassero più cifre decimali, sarebbe più semplice individuare quale dei 5 algoritmi genera il modello migliore (con valori di indici più bassi). Di seguito, nella *tab.8* sono riportati i migliori modelli generati dai relativi algoritmi ad ogni simulazione effettuata, con annessi i valori ottimi dei parametri *a(1)*, *a(2)* e *a(3)*.

SIMULAZ	nr. ITER	nr. PART	ALGORITMO migliore	*a(1)*	*a(2)*	*a(3)*
1	25	25	FFA	-0,0292	-0,8789	0,9997
2	25	50	FFA	-0,0293	-0,8794	1,0000
3	25	75	FFA	-0,0293	-0,8796	1,0000
4	25	100	FFA	-0,0285	-0,8796	0,9990
5	50	25	FFA	-0,0294	-0,8794	1,0000
6	50	50	FFA	-0,0287	-0,8792	0,9992
7	50	75	FFA	-0,0287	-0,8795	0,9992
8	50	100	FFA	-0,0285	-0,8791	0,9990
9	75	25	FFA	-0,0283	-0,8792	0,9987
10	75	50	FFA	-0,0285	-0,8794	0,9990
11	75	75	PSO	-0,0283	-0,8794	0,9988
12	75	100	FFA	-0,0285	-0,8794	0,9990
13	100	25	PSO	-0,0286	-0,8794	0,9991
14	100	50	PSO	-0,0285	-0,8794	0,9991
15	100	75	PSO	-0,0285	-0,8794	0,9991
16	100	100	PSO	-0,0285	-0,8794	0,9991

tab.8

Dalla *tab.8* si evince che al variare del numero di particelle e di iterazioni, i valori ottimi da attribuire ai parametri rimangono praticamente quasi gli stessi, variando in maniera minima. Per quanto riguarda invece la tipologia migliore di algoritmo (seppur i due forniscono risultati molto simili come detto precedentemente), aumentando il numero di iterazioni sarebbe più opportuno utilizzare il modello fornito dall'algoritmo PSO, piuttosto che quello fornito dal FFA.

Effettuando un'analisi più qualitativa, andando quindi ad esaminare le *Convergence_curve* di ciascun algoritmo, si possono confrontare le diverse velocità di convergenza ed i valori minimi di MSE che gli stessi riescono a garantire. In *fig.45* e *fig.46* sono riportate le *Convergence_curve* risultanti dalle simulazioni effettuate, considerando nel primo caso 25 iterazioni – 75 particelle e nel secondo caso 75 iterazioni – 50 particelle.

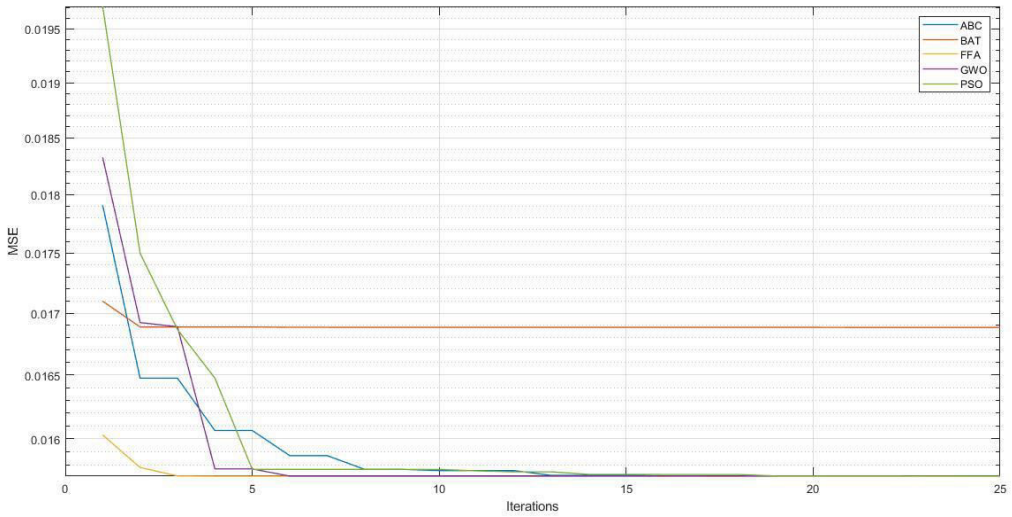


fig.45

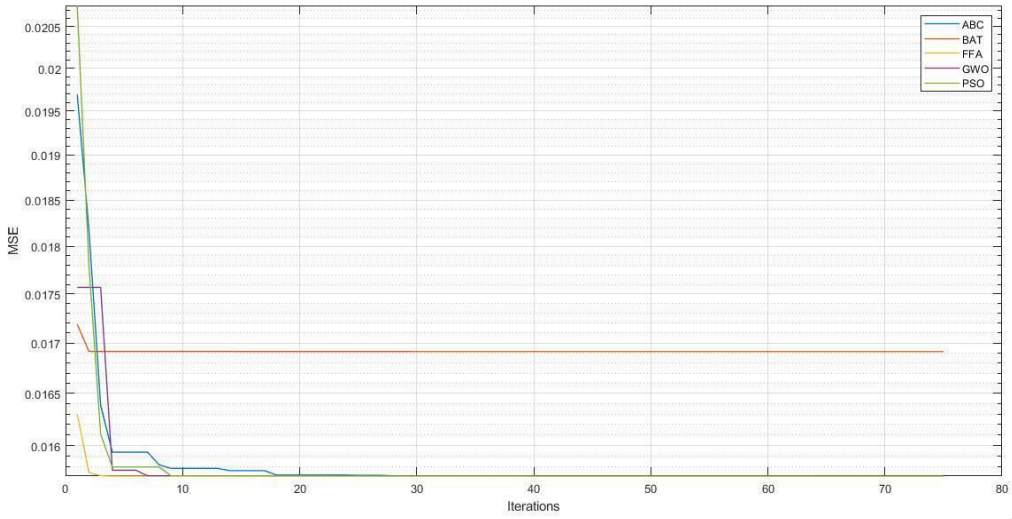


fig.46

In entrambi i casi si può osservare che il FFA risulta avere una velocità di convergenza migliore (più elevata) verso il valore minimo di MSE, rispetto agli altri 4 algoritmi. Il BAT, pur presentando una convergenza abbastanza elevata (entro le prime 10 iterazioni), converge ad un valore di MSE maggiore e risulta pertanto essere il peggiore come indicato anche in precedenza. All'incirca l'ABC, il FFA, il GWO e il PSO convergono ad un MSE pari a 0,016. Tale valore risulta superiore all'MSE minimo riscontrato sia considerando l'equazione parametrica **lineare** (0,004), che l'equazione parametrica **quadratica** (0,009) e fornisce quindi un' ulteriore conferma della migliore bontà del legame **lineare**. Tutto ciò è stato riscontrato anche per gli altri valori di iterazioni e particelle.

Conclusioni

Questo progetto di tesi ha voluto inizialmente illustrare sia l'importanza dei sistemi Digital Twin e di come essi possano generare vantaggi competitivi, sia l'importanza dell'utilizzo delle tecniche di apprendimento automatico. Nel contesto di Industry 4.0 questi due concetti si sposano perfettamente e permettono la creazione di vari modelli digitali di macchinari, processi e servizi. Lo studio è stato condotto esaminando un eiettore il cui modello fisico non era noto a priori. Utilizzando algoritmi di apprendimento automatico, basati sulla Swarm Intelligence e sfruttando dei dati sperimentali acquisiti da varie misurazioni sul campo, è stato possibile ricostruire un modello digitale (Digital Twin) del dispositivo stesso. E' stata quindi effettuata un'identificazione parametrica del dispositivo, al fine di individuare un'equazione in grado di caratterizzarne il comportamento. Dalle analisi svolte è stato riscontrato che l'equazione parametrica che meglio rappresenta il modello dell'eiettore, tra quelle proposte, risulta essere quella lineare. Tali risultati sono stati ottenuti considerando solo determinate tipologie di dati sperimentali, tra quelle presenti. Il modello ricavato infatti rappresenta l'eiettore nella configurazione attuale, ossia con dimensione degli ugelli D1 e D2 pari a 11mm e 29mm. Se si volesse creare un modello più preciso o inerente ad altre configurazioni, si potrebbero ipotizzare legami/equazioni parametriche diverse, sfruttare più tipologie di dati sperimentali e considerare anche altre possibili dimensioni degli ugelli.

Appendice A

```
function [BestSol_PSO,Convergence_curve_PSO,output_stimata_PSO]=PSO …
(nPop,Max_iter,lb,ub,dim,input,output,type_fun)

    VarSize=[1 dim];      % Size of Decision Variables Matrix
    VarMin=lb(1);         % Lower Bound of Variables
    VarMax=ub(1);         % Upper Bound of Variables

    %% PSO Parameters

    w=1;                  % Inertia Weight
    wdamp=0.99;           % Inertia Weight Damping Ratio
    c1=1.5;               % Personal Learning Coefficient
    c2=2.0;               % Global Learning Coefficient

    % Velocity Limits
    VelMax=0.1*(VarMax-VarMin);
    VelMin=-VelMax;

    %% Initialization

    empty_particle.Position=[];
    empty_particle.Cost=[];
    empty_particle.Velocity=[];
    empty_particle.Best.Position=[];
    empty_particle.Best.Cost=[];

    particle=repmat(empty_particle,nPop,1);

    GlobalBest.Cost=inf;

    for i=1:nPop

        % Initialize Position
        particle(i).Position=unifrnd(VarMin,VarMax,VarSize);

        % Initialize Velocity
        particle(i).Velocity=zeros(VarSize);

        % Evaluation
        particle(i).Cost=calcola_costo(particle(i).Position,input,output,type_fun);

        % Update Personal Best
        particle(i).Best.Position=particle(i).Position;
        particle(i).Best.Cost=particle(i).Cost;

        % Update Global Best
        if particle(i).Best.Cost<GlobalBest.Cost
```

```
                GlobalBest=particle(i).Best;

        end

end

Convergence_curve_PSO=zeros(Max_iter,1);

%% PSO Main Loop

for it=1:Max_iter

        for i=1:nPop

                % Update Velocity
                particle(i).Velocity = w*particle(i).Velocity ...
                +c1*rand(VarSize).*(particle(i).Best.Position-particle(i).Position) ...
                +c2*rand(VarSize).*(GlobalBest.Position-particle(i).Position);

                % Apply Velocity Limits
                particle(i).Velocity = max(particle(i).Velocity,VelMin);
                particle(i).Velocity = min(particle(i).Velocity,VelMax);

                % Update Position
                particle(i).Position = particle(i).Position + particle(i).Velocity;

                % Velocity Mirror Effect
                IsOutside=(particle(i).Position<VarMin | particle(i).Position>VarMax);
                particle(i).Velocity(IsOutside)=-particle(i).Velocity(IsOutside);

                % Apply Position Limits
                particle(i).Position = max(particle(i).Position,VarMin);
                particle(i).Position = min(particle(i).Position,VarMax);

                % Evaluation
                particle(i).Cost=calcola_costo …
                (particle(i).Position,input,output,type_fun);

                % Update Personal Best
                if particle(i).Cost<particle(i).Best.Cost

                        particle(i).Best.Position=particle(i).Position;
                        particle(i).Best.Cost=particle(i).Cost;

                        % Update Global Best
                        if particle(i).Best.Cost<GlobalBest.Cost

                                GlobalBest=particle(i).Best;

                        end
```

```
                end

            end

            Convergence_curve_PSO(it)=GlobalBest.Cost;

            w=w*wdamp;

        end

        BestSol_PSO = GlobalBest;

        %% Results

        [costo,output_stimata_PSO]=calcola_costo …
        (BestSol_PSO.Position,input,output,type_fun);

end
```

Bibliografia

Ab Wahab, Mohd Nadhir, Samia Nefti-Meziani, e Adham Atyabi. 2015a. «A Comprehensive Review of Swarm Optimization Algorithms» a cura di C. Buiu. *PLOS ONE* 10(5):e0122827.

Ab Wahab, Mohd Nadhir, Samia Nefti-Meziani, e Adham Atyabi. 2015b. «A Comprehensive Review of Swarm Optimization Algorithms» a cura di C. Buiu. *PLOS ONE* 10(5):e0122827.

Ab Wahab, Mohd Nadhir, Samia Nefti-Meziani, e Adham Atyabi. 2015c. «A Comprehensive Review of Swarm Optimization Algorithms» a cura di C. Buiu. *PLOS ONE* 10(5):e0122827.

Abdallah, Hazem, Hassan M. Emara, Hassan T. Dorrah, e Ahmed Bahgat. 2009. «Using Ant Colony Optimization Algorithm for Solving Project Management Problems». *Expert Systems with Applications* 36(6):10004–15.

Akhtar, S., A. R. Ahmad, e E. M. Abdel-Rahman. 2012. «A Metaheuristic Bat-Inspired Algorithm for Full Body Human Pose Estimation». Pagg. 369–75 in *2012 Ninth Conference on Computer and Robot Vision*. Toronto, Ontario, Canada: IEEE.

Al-anzi, Fawaz, e Ali Allahverdi. 2006. «Computer Assembly Scheduling Problem». Pagg. 1188–91 in *2006 International Conference on Service Systems and Service Management*. Troyes, France: IEEE.

Bhargava, V., S. E. K. Fateen, e A. Bonilla-Petriciolet. 2013. «Cuckoo Search: A New Nature-Inspired Optimization Method for Phase Equilibrium Calculations». *Fluid Phase Equilibria* 337:191–200.

Bishop, Chris M. 1994. «Neural Networks and Their Applications». *Neural Networks* 65(6):30.

Blum, Christian, e Andrea Roli. 2003. «Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison». *ACM Computing Surveys* 35(3):41.

Boschert, Stefan, e Roland Rosen. 2016. «Digital Twin—The Simulation Aspect». Pagg. 59–74 in *Mechatronic Futures*, a cura di P. Hehenberger e D. Bradley. Cham: Springer International Publishing.

Burnham, Kenneth P., e David R. Anderson. 2004. «Multimodel Inference: Understanding AIC and BIC in Model Selection». *Sociological Methods & Research* 33(2):261–304.

Chevuru, Ramesh, e G. V. Martheswar. 2014. «Implementation of a New Methodology for ELD Problems». 2(10):7.

Chidambaram, Chidambaram, e Heitor Silverio Lopes. 2009. «A New Approach for Template Matching in Digital Images Using an Artificial Bee Colony Algorithm». Pagg. 146–51 in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. Coimbatore: IEEE.

Chunnanond, Kanjanapon, e Satha Aphornratana. 2004. «Ejectors: Applications in Refrigeration Technology». *Renewable and Sustainable Energy Reviews* 8(2):129–55.

Colin, J. Parris. 2016. «Mind + Machines: Meet a Digital Twin».

Das, Kaushik Ranjan, M. Tech, Diptanu Das, e Joyashree Das. 2015. «Optimal Tuning of PID Controller Using GWO Algorithm for Speed Control in DC Motor». 5.

Eberhart, R. C., e Xiaohui Hu. 1999. «Human Tremor Analysis Using Particle Swarm Optimization». Pagg. 1927–30 in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Washington, DC, USA: IEEE.

Elbel, Stefan. 2011. «Historical and Present Developments of Ejector Refrigeration Systems with Emphasis on Transcritical Carbon Dioxide Air-Conditioning Applications». *International Journal of Refrigeration* 34(7):1545–61.

Fister, Iztok, Iztok Fister, Xin-She Yang, e Janez Brest. 2013. «A Comprehensive Review of Firefly Algorithms». *Swarm and Evolutionary Computation* 13:34–46.

Fister Jr., Iztok, Xin-She Yang, Iztok Fister, Janez Brest, e Dušan Fister. 2013. «A Brief Review of Nature-Inspired Algorithms for Optimization». *ArXiv:1307.4186 [Cs]*.

Fuellerer, Guenther, Karl F. Doerner, Richard F. Hartl, e Manuel Iori. 2009. «Ant Colony Optimization for the Two-Dimensional Loading Vehicle Routing Problem». *Computers & Operations Research* 36(3):655–73.

Galanis, Nicolas, e Mikhail Sorin. 2016. «Ejector Design and Performance Prediction». *International Journal of Thermal Sciences* 104:315–29.

Gallo, Crescenzio. 2007. «Reti Neurali Artificiali: Teoria ed Applicazioni Finanziarie». (28):1–47.

Gandomi, Amir Hossein, Xin-She Yang, e Amir Hossein Alavi. 2013. «Cuckoo Search Algorithm: A Metaheuristic Approach to Solve Structural Optimization Problems». *Engineering with Computers* 29(1):17–35.

Gay, Norman H. 1931. «Refrigerating System».

Gelfusa, Davide. 2013. «Tecniche di ottimizzazione per la stima dei parametri di modelli dinamici lineari». UNIVERSITÀ DEGLI STUDI DI ROMA TOR VERGATA.

Gorai, Apurba, e Ashish Ghosh. 2011. «Hue-Preserving Color Image Enhancement Using Particle Swarm Optimization». 6.

Grigolo, Fabio. 2013. «Analisi e implementazione di algoritmi e strategie per l'ottimizzazione del taglio con cesoia (nesting rettangolare)». Università degli Studi di Padova.

Hamad, Faten. 2018. «Using Artificial Bee Colony Algorithm for Test Data Generation and Path Testing Coverage». *Modern Applied Science* 12(7):99.

Haoran, Chen, e Cai Wenjian. 2014. «Artificial Neural Network Modeling for Variable Area Ratio Ejector». Pagg. 220–25 in *2014 9th IEEE Conference on Industrial Electronics and Applications*. Hangzhou, China: IEEE.

Horng, Ming-Huwi, e Ting-Wei Jiang. 2010. «The Codebook Design of Image Vector Quantization Based on the Firefly Algorithm». Pagg. 438–47 in *Computational Collective Intelligence. Technologies and Applications*. Vol. 6423, *Lecture Notes in Computer Science*, a cura di J.-S. Pan, S.-M. Chen, e N. T. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg.

Hu, Hui. 2012. «FA-Based Optimal Strategy of Train's Energy Saving with Energy Materials». *Advanced Materials Research* 485:93–96.

Huang, B. J., J. M. Chang, C. P. Wang, e V. A. Petrenko. 1999. «A 1-D Analysis of Ejector Performance». *International Journal of Refrigeration* 22(5):354–64.

Jun, Zhang, e Zhang Kanyu. 2011. «A Particle Swarm Optimization Approach for Optimal Design of PID Controller for Temperature Control in HVAC». Pagg. 230–33 in *2011 Third International Conference on Measuring Technology and Mechatronics Automation*. Shanghai, China: IEEE.

Karaboga, Dervis, Selcuk Okdem, e Celal Ozturk. 2010. «Cluster Based Wireless Sensor Network Routings Using Artificial Bee Colony Algorithm». Pagg. 1–5 in *2010 International Conference on Autonomous and Intelligent Systems, AIS 2010*. Povoa de Varzim, Portugal: IEEE.

Kim, Myoung Il, Og Sin Kim, Dong Hyun Lee, e Sang Done Kim. 2007. «Numerical and Experimental Investigations of Gas–Liquid Dispersion in an Ejector». *Chemical Engineering Science* 62(24):7133–39.

Komaki, G. M., e Vahid Kayvanfar. 2015. «Grey Wolf Optimizer Algorithm for the Two-Stage Assembly Flow Shop Scheduling Problem with Release Time». *Journal of Computational Science* 8:109–20.

Kordon, Arthur K. 2010. «Swarm Intelligence: The Benefits of Swarms». Pagg. 145–74 in *Applying Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kranakis, Eda Fowlks. 1982. «The French Connection: Giffard's Injector and the Nature of Heat». *Technology and Culture* 23(1):3.

Kumar, Anil, e Shampa Chakarverty. 2011. «Design Optimization for Reliable Embedded System Using Cuckoo Search». Pagg. 264–68 in *2011 3rd International Conference on Electronics Computer Technology*. Kanyakumari, India: IEEE.

Kumar, S. Barath, e G. Myilsamy. s.d. «Multi-Target Tracking in Mobility Sensor Networks Using Ant Colony Optimization». 5.

Li, C., e Y. Z. Li. 2011. «Investigation of Entrainment Behavior and Characteristics of Gas–Liquid Ejectors Based on CFD Simulation». *Chemical Engineering Science* 66(3):405–16.

Liu, Jiapeng, Lei Wang, e Lei Jia. 2017. «A Predictive Model for the Performance of the Ejector in Refrigeration System». *Energy Conversion and Management* 150:269–76.

Liu, Tao, Ai-ling Qi, Yuan-bin Hou, e Xin-tan Chang. 2009. «Feature Optimization Based on Artificial Fish-Swarm Algorithm in Intrusion Detections». Pagg. 542–45 in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*. Wuhan, China: IEEE.

Marzano, Luca. 2017. «Algoritmi di Deep Learning dell'analisi di dati scientifici». Università degli Studi di Bari - Aldo Moro.

Mazzelli, Federico, Francesco Giacomelli, e Adriano Milazzo. 2018. «CFD Modeling of Condensing Steam Ejectors: Comparison with an Experimental Test-Case». *International Journal of Thermal Sciences* 127:7–18.

Mazzotti, Matteo. 2008. «Formulazione agli elementi finiti di problemi di identificazione parametrica». ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA.

Mingyan Jiang, e Dongfeng Yuan. 2005. «Wavelet Threshold Optimization with Artificial Fish Swarm Algorithm». Pagg. 569–72 in *2005 International Conference on Neural Networks and Brain*. Vol. 1. Beijing, China: IEEE.

Mirjalili, Seyedali, Seyed Mohammad Mirjalili, e Andrew Lewis. 2014. «Grey Wolf Optimizer». *Advances in Engineering Software* 69:46–61.

Moravej, Zahra, e Amir Akhlaghi. 2013. «A Novel Approach Based on Cuckoo Search for DG Allocation in Distribution Network». *International Journal of Electrical Power & Energy Systems* 44(1):672–79.

Narimani, Elhameh, Mikhail Sorin, Philippe Micheau, e Hakim Nesreddine. 2019. «Dynamic Modeling of an R245fa Ejector Based Refrigeration System». *International Journal of Refrigeration* 107:262–74.

Nasseh, S., A. Mohebbi, Z. Jeirani, e A. Sarrafi. 2007. «Predicting Pressure Drop in Venturi Scrubbers with Artificial Neural Networks». *Journal of Hazardous Materials* 143(1–2):144–49.

Negri, Elisa, Luca Fumagalli, e Marco Macchi. 2017. «A Review of the Roles of Digital Twin in CPS-Based Production Systems». *Procedia Manufacturing* 11:939–48.

Neshat, Mehdi, Ali Adeli, Ghodrat Sepidnam, Mehdi Sargolzaei, e Adel Najaran Toosi. 2012. «A REVIEW OF ARTIFICIAL FISH SWARM OPTIMIZATION METHODS AND APPLICATIONS». *International Journal on Smart Sensing and Intelligent Systems* 5(1):108–48.

Nguyen, Trong-The, Ho Thi Huong Thom, e Thi-Kien Dao. 2017. «Estimation Localization in Wireless Sensor Network Based on Multi-Objective Grey Wolf Optimizer». Pagg. 228–37 in *Advances in Information and Communication Technology*. Vol. 538, *Advances in Intelligent Systems and Computing*, a cura di M. Akagi, Thanh-Thuy Nguyen, D.-T. Vu, T.-N. Phung, e V.-N. Huynh. Cham: Springer International Publishing.

Pansuwan, Primpika, Niyada Rukwong, e Pupong Pongcharoen. 2010. «Identifying Optimum Artificial Bee Colony (ABC) Algorithm's Parameters for Scheduling the Manufacture and Assembly of Complex Products». Pagg. 339–43 in *2010 Second International Conference on Computer and Network Technology*. Bangkok, Thailand: IEEE.

Priano, Niccolò. 2013. «Studio di prefattibilità per il recupero e la valorizzazione del gas di torcia su piattaforme offshore». Politecnico di Milano.

Qianzhi Ma, e Xiujuan Lei. 2010. «Application of Artificial Fish School Algorithm in UCAV Path Planning». Pagg. 555–59 in *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*. Changsha, China: IEEE.

Raftery, Adrian E. 1995. «Bayesian Model Selection in Social Research». *Sociological Methodology* 25:111.

Rao, R. Srinivasa. 2010. «Capacitor Placement in Radial Distribution System for Loss Reduction Using Artificial Bee Colony Algorithm». *International Journal of Electronics and Communication Engineering* 4(8):5.

Reynolds, Andy M., e Mark A. Frye. 2007. «Free-Flight Odor Tracking in Drosophila Is Consistent with an Optimal Intermittent Scale-Free Search» a cura di S. Rands. *PLoS ONE* 2(4):e354.

Riccò, Fabio. 2013. «Stima del rendimento effettivo (Performance Ratio) di moduli fotovoltaici con e senza modelli matematici». Politecnico di Milano.

Rizzo, Michael. 2018. «PSO e BFO per la gestione di hedge funds: il caso DeltaHedge». Università Ca' Foscari - Venezia.

Saka, M. P., E. Doğan, e Ibrahim Aydogdu. 2013. «Analysis of Swarm Intelligence–Based Algorithms for Constrained Optimization». Pagg. 25–48 in *Swarm Intelligence and Bio-Inspired Computation*. Elsevier.

Saracco, Roberto. 2018. «Industry 4.0, come il modello "digital twin" migliora sviluppo e prodotti». *Agenda Digitale*.

Schluse, Michael, e Juergen Rossmann. 2016. «From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems». Pagg. 1–6 in *2016 IEEE International Symposium on Systems Engineering (ISSE)*. Edinburgh, United Kingdom: IEEE.

Senthilnath, J., S. N. Omkar, e V. Mani. 2011. «Clustering Using Firefly Algorithm: Performance Study». *Swarm and Evolutionary Computation* 1(3):164–71.

Song, Xianhai, Li Tang, Sutao Zhao, Xueqiang Zhang, Lei Li, Jianquan Huang, e Wei Cai. 2015. «Grey Wolf Optimizer for Parameter Estimation in Surface Waves». *Soil Dynamics and Earthquake Engineering* 75:147–57.

Sözen, Adnan, e Erol Arcaklioğlu. 2007. «Exergy Analysis of an Ejector-Absorption Heat Transformer Using Artificial Neural Network Approach». *Applied Thermal Engineering* 27(2–3):481–91.

Taixiong Zheng, e Jiongqiu Li. 2010. «Multi-Robot Task Allocation and Scheduling Based on Fish Swarm Algorithm». Pagg. 6681–85 in *2010 8th World Congress on Intelligent Control and Automation*. Jinan, China: IEEE.

Tashtoush, Bourhan M., Moh'd A. Al-Nimr, e Mohammad A. Khasawneh. 2019. «A Comprehensive Review of Ejector Design, Performance, and Applications». *Applied Energy* 240:138–72.

Tu, Jack V. 1996. «Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes». *Journal of Clinical Epidemiology* 49(11):1225–31.

Udaiyakumar, K. C., e M. Chandrasekaran. 2014. «Application of Firefly Algorithm in Job Shop Scheduling Problem for Minimization of Makespan». *Procedia Engineering* 97:1798–1807.

Valdez, Fevrier, e Ivan Chaparro. 2013. «Ant Colony Optimization for Solving the TSP Symetric with Parallel Processing». Pagg. 1192–96 in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. Edmonton, AB, Canada: IEEE.

Wang, Hao, Wenjian Cai, e Youyi Wang. 2016. «Modeling of a Hybrid Ejector Air Conditioning System Using Artificial Neural Networks». *Energy Conversion and Management* 127:11–24.

Wang, Jie-Sheng, e Shu-Xia Li. 2019. «An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism». *Scientific Reports* 9(1):7181.

Yang, Xin She, e Xingshi He. 2013. «Bat Algorithm: Literature Review and Applications». *International Journal of Bio-Inspired Computation* 5(3):141.

Yang, Xin-She. 2009. «Firefly Algorithms for Multimodal Optimization». Pagg. 169–78 in *Stochastic Algorithms: Foundations and Applications*. Vol. 5792, *Lecture Notes in Computer Science*, a cura di O. Watanabe e T. Zeugmann. Berlin, Heidelberg: Springer Berlin Heidelberg.

Yang, Xin-She, Mehmet Karamanoglu, e Simon Fong. 2012. «Bat Algorithm for Topology Optimization in Microelectronic Applications». Pagg. 150–55 in *The First International Conference on Future Generation Communication Technologies*. London, United Kingdom: IEEE.

Yildiz, Ali R. 2013. «Cuckoo Search Algorithm for the Selection of Optimal Machining Parameters in Milling Operations». *The International Journal of Advanced Manufacturing Technology* 64(1–4):55–61.

Zedadra, Ouarda, Antonio Guerrieri, Nicolas Jouandeau, Giandomenico Spezzano, Hamid Seridi, e Giancarlo Fortino. 2018. «Swarm Intelligence-Based Algorithms within IoT-Based Systems: A Review». *Journal of Parallel and Distributed Computing* 122:173–87.

Zheng, Ping, Bing Li, e Jingxuan Qin. 2018. «CFD Simulation of Two-Phase Ejector Performance Influenced by Different Operation Conditions». *Energy* 155:1129–45.

Zhu, Yinhai, Wenjian Cai, Changyun Wen, e Yanzhong Li. 2007. «Shock Circle Model for Ejector Performance Evaluation». *Energy Conversion and Management* 48(9):2533–41.

Zuccon, Matteo. 2017. «Applicazione della Particle Swarm Optimization per problemi di Tracking Error portfolio». Università Ca' Foscari - Venezia.

Ringraziamenti

Al termine di questo progetto di tesi, desidero ringraziare diverse persone che, in un modo o nell'altro, sono state fondamentali, in quanto mi hanno permesso di raggiungere questo traguardo.

In primo luogo, vorrei ringraziare il Prof. Maurizio Bevilacqua per avermi offerto la possibilità di affrontare argomenti innovativi e ricchi di applicazioni reali, e per avermi dato l'occasione di approcciarmi, dal punto di vista sperimentale, ad un progetto importante sfruttando le risorse offerte dal Dipartimento.

Desidero ringraziare infinitamente anche il Prof. Giovanni Mazzuto per avermi aiutato in maniera determinante alla realizzazione di questo progetto. La sua disponibilità e la sua collaborazione sono state essenziali. Lo ringrazio ancora per la pazienza dimostrata e per tutti i consigli fornitimi.

Un ringraziamento veramente speciale va alla mia famiglia. Vi ringrazio per tutti gli sforzi e i sacrifici fatti, per avermi sostenuto sia moralmente che economicamente. Senza di voi nulla di tutto ciò sarebbe stato possibile. Mi avete dato la possibilità di scegliere la strada che volevo, permettendomi di costruire il mio futuro. Grazie davvero per aver sempre creduto in me.

Ringrazio anche alcune persone care che mi hanno cresciuto, ma che purtroppo non ci sono più: i miei nonni Amalio, Fiorisa e Primo. Avete contribuito in maniera fondamentale allo sviluppo della persona che sta scrivendo queste parole. Grazie veramente di cuore.

Ringrazio anche i miei zii, cugini e parenti, che sono stati sempre pronti a gioire dei miei traguardi, dimostrandomi il loro affetto.

Desidero ora ringraziare una persona davvero speciale, una persona che in questi tre anni mi è stata sempre accanto. Ebbene sì Meri, sto parlando proprio di te. La nostra relazione è nata esattamente nei primi giorni in cui iniziavo ad intraprendere questo percorso universitario. Da quel momento per me ci sei stata sempre. Sei stata, e sei, una fidanzata stupenda, in tutti i sensi. Grazie per avermi sempre supportato e sopportato in tutto quel che ho fatto, grazie per i mille consigli dati, grazie per essere stata una fonte inesauribile di incoraggiamento e per avermi dato sempre la giusta spinta e motivazione nei momenti più difficili. Grazie semplicemente, per tutto l'amore che mi ha dimostrato.

Ringrazio in modo particolare anche te, Gina, per avermi trattato come un figlio. Grazie per l'affetto dimostratomi e per i mille pranzi e le mille cene offerte.

Ringrazio tutta la "FermoCityGang": i miei compagni universitari conosciuti in quel di Fermo. Insieme abbiamo condiviso gioie e "dolori" senza mai arrenderci, ma anche bellissime serate passate nei vari pub e ristoranti di Fermo. È anche grazie al vostro supporto se finalmente ho raggiunto questo traguardo.

Ringrazio Metti, Eug, Lepo, Joseph e Won, i miei fantastici coinquilini, che in questi tre anni mi hanno fatto passare piacevoli momenti di quotidianità. Grazie a loro la vita da fuorisede è stata meno dura del previsto, anzi a dire la verità è stata proprio uno "scialo"!

Vorrei spendere due parole in più per te Metti e per te Eug. Per me siete diventati come fratelli; sempre pronti ad aiutarci e ad incoraggiarci a vicenda nei momenti di difficoltà e sempre pronti a festeggiare i successi di ognuno. Adesso che questo percorso universitario è giunto al termine, spero che il nostro legame continui ad essere forte, nonostante le nuove strade che percorreremo.

Per ultimi, ma non meno importanti, desidero ringraziare tutti i miei amici di Poggio (e dintorni) con i quali passo il mio tempo libero, e che non vedono l'ora di festeggiare con me questa tappa così importante della mia vita. Vi ringrazio per farmi vivere continuamente momenti di spensieratezza e felicità in compagnia. Avete contribuito anche voi e non poco al raggiungimento di questo mio traguardo.

In particolare vorrei ringraziare Reg, Dami, Clà, Jona e Stecco: i miei "Parassiti". Grazie per aver vissuto insieme a me, momenti ed esperienze indimenticabili… chi se lo scorda più il caffè preso a Roma alle 6:00 di mattina, o la partenza in salita con la Zafira di Stecco a Peschici! Vi ringrazio di essere quello che siete: amici unici, divertenti e … ritardatari!

Grazie ancora a tutti.

Jacopo