UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Dipartimento Ingegneria Industriale e Scienze Matematiche – DIISM

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCANICA
CURRICULUM MECCATRONICA

Applicazione di algoritmi Swarm Intelligence per la predizione della solubilità dell'anidride carbonica nei processi di Carbon Capture

Application of Swarm Intelligence algorithms for predicting carbon dioxide solubility in carbon capture processes

Relatore:
Prof. Giovanni Mazzuto

Tesi di laurea di:
Tommaso Canullo

Anno accademico 2023/2024

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Meccanica

INDICE

1 Introduzione

La tesi in questione ha come obiettivo la predizione di valori di Solubilità (S) dell'anidride carbonica (CO_2), all'interno di soluzioni con Liquidi Ionici (Ionic Liquids, IL), in diverse condizioni di Temperatura (T) e Pressione (P). Esso si colloca nel settore della ricerca per l'ambiente, dell'ottimizzazione degli impianti meccanici e dell'innovazione nella ricerca e nell'utilizzo dei materiali.

Il progetto, infatti, trova la sua ragione di esistere nel mondo del Carbon Capture and Storage (CCS), ad oggi una delle principali strade da percorrere per riuscire a ridurre le emissioni inquinanti, e tutti i problemi che ne derivano.

Le emissioni antropogeniche di CO_2 e altri gas serra, infatti, sono la principale causa del riscaldamento globale e dei cambiamenti climatici. Le concentrazioni di anidride carbonica nell'atmosfera sono aumentate da 280 ppm a metà del diciannovesimo secolo, a circa 404 ppm nel 2016, causando un aumento di quasi 1 °C della temperatura media terrestre rispetto ai livelli preindustriali. A sua volta, questo incremento di temperatura ha comportato un aumento del livello medio del mare di 20 cm, tra il 1901 e il 2010 (Castro-Pardo et al., 2022).

Successivamente, l'inquinamento globale ha raggiunto i livelli più alti mai registrati nella storia umana. Per affrontare questa situazione è stato approvato l'Accordo di Parigi, proposto alla ventunesima conferenza delle Nazioni Unite sui cambiamenti climatici. Si è fissato un limite per il controllo della temperatura media globale, a 2 °C al di sotto dei livelli preindustriali. Un altro provvedimento importante è stato il sesto rapporto di valutazione dell'Intergovernmental Panel sui cambiamenti climatici, che ha proposto che le emissioni globali di gas serra debbano raggiungere il picco prima del 2025, e che si debba ottenere una riduzione del 43% delle emissioni entro il 2030 per limitare il riscaldamento a 1,5 °C, mentre le emissioni dovranno ridursi del 25% entro il 2030 per limitare il riscaldamento a 2 °C (Yao et al., 2023).

Le tecnologie di cattura della CO_2 sono, quindi, di importanza globale per poter raggiungere questi obiettivi, poichè le emissioni derivanti da combustibili fossili stanno minacciando significativamente l'economia, l'ambiente, gli ecosistemi naturali e la salute umana. I progressi nella tecnologia di cattura e utilizzo del carbonio possono ridurre drasticamente l'inquinamento, e guidare il percorso verso zero emissioni nette, raggiungendo anche diversi obiettivi di sviluppo sostenibile delle Nazioni Unite: in particolare l'Obiettivo 6 (Acqua pulita e servizi igienico-sanitari), l'Obiettivo 7 (Energia pulita e accessibile), l'Obiettivo 9 (Industria, innovazione e infrastrutture), l'Obiettivo 11 (Città e comunità sostenibili), l'Obiettivo 12 (Consumo e produzione responsabili) e l'Obiettivo 13 (Azione per il clima) (Podder et al., 2023).

Diventa, quindi, fondamentale poter modellare questo tipo di impianti e processi nella miglior maniera possibile, per apportarvi sempre più innovazione e qualità, col fine di ottenere un mondo migliore. Per far ciò, uno degli strumenti più potenti sono sicuramente tutte le tecniche di Intelligenza Artificiale (Artificial Intelligence, AI), che permettono di gestire, con un controllo profondo, ogni tipo di applicazione e processo, affermandosi come uno dei pilastri di indutria 4.0 e 5.0. In questa trattazione, nello specifico, la previsione dei valori di S viene eseguita proprio tramite lo sviluppo e la personalizzazione di un algoritmo di AI, che verrà analizzato in seguito.

La tesi è strutturata con due capitoli introduttivi posti prima di quello principale, che espone il progetto nella sua interezza. Il capitolo Tecnologie di Carbon Capture propone una panoramica illustrativa su tutte le principali tecniche CCS ad oggi utilizzate, iniziando con dei cenni storici riguardanti i primi impianti realizzati, e quelli ad oggi esistenti; proseguendo con uno stato dell'arte che mostra nel dettaglio i diversi tipi di processi di cattura del carbonio.

Per la stesura di questa trattazione, la ricerca è stata effettuata sul database bibliografico Scopus, analizzando la letteratura e ricavando idee ed informazioni utili al progetto. In particolare, vengono riportate le chiavi utilizzate per la ricerca. Nel primo capitolo quella più importante è stata, chiaramente, “Carbon Capture”; che, per la parte storica e introduttiva sugli impianti CCS, è stata abbinata a “First”, oppure “Early”, “History” o “Plant”. Successivamente, dovendo fare una panoramica delle principali tecnologie, si è pensato di abbinare la chiave principale a “Overview” o “Review”, insieme a “Technologies”.

Di seguito, il capitolo Algoritmi di Intelligenza Artificiale per la modellazione dei processi di Carbon Capture, parte con l’illustrare le principali applicazioni del Machine Learning (ML) e delle tecniche di AI in contesto industriale, e segue approfondendo, come dice il titolo, relativamente al settore CCS. Le chiavi utilizzate per questa sezione sono state, oltre a “Carbon Capture”, anche "Modeling", insieme ad "AI" oppure "Algorithm", "Intelligence" o, ancora, "Machine Learning".

Infine, verrà mostrato il percorso fatto all’interno di Materiali e Metodi, dove si illustrano le parti principali dell’algoritmo sviluppato. Una volta fatto ciò, saranno esposti ed analizzati i risultati ottenuti, prima di trarre le conclusioni finali. Per i primi due paragrafi di materiali e metodi, la ricerca è stata semplicemente abbinare, rispettivamente, le chiavi “Swarm Intelligence” e “GWO” o “Grey Wolf Optimizer”, con “Review” oppure “Overview”.

2 Tecnologie di Carbon Capture

Le tecnologie di cattura e stoccaggio del carbonio hanno una lunga storia, risalente agli anni '20, quando venivano utilizzate per separare la CO_2 dal metano, nei giacimenti di gas naturale. Più avanti, negli Stati Uniti degli anni '70, la CO_2 veniva estratta, trasportata e iniettata nei giacimenti petroliferi, per aumentare la produzione di petrolio, un processo noto come Enhanced Oil Recovery (EOR). Un esempio pionieristico ne è il progetto Scurry Area Canyon Reef Operating Committee (SACROC), avviato nel 1972 da Chevron, in Texas. In questo progetto, durante il periodo 1972-2009, sono state iniettate oltre 175 milioni di tonnellate di CO_2 naturale. Inizialmente, quindi, il CCS veniva utilizzato per scopi commerciali, e solo successivamente è diventato una soluzione interessante per ridurre le emissioni di gas serra e mitigare il cambiamento climatico.

L'idea di catturare, trasportare e stoccare la CO_2 per ridurre le emissioni antropogeniche è stata proposta per la prima volta da Cesare Marchetti, nel 1977. Egli propose di utilizzare i processi di separazione della CO_2 già impiegati nell'industria, per purificare l'idrogeno ottenuto dal metano, integrandoli direttamente all'interno delle centrali elettriche. Successivamente, il concetto di CCS moderno ha cominciato a prender forma con progetti come il Sleipner CCS e il Weyburn-Midale CO_2 Monitoring and Storage Project.

Il progetto Sleipner, avviato nel 1996, prevedeva la cattura della CO_2 separata nel processo di purificazione del gas naturale e la sua iniezione in acquiferi salini profondi. Fino ad oggi, il progetto ha accumulato oltre 20 milioni di tonnellate di CO_2 stoccata.

Il progetto Weyburn, iniziato nel 2000, è il più completo progetto di ricerca multidisciplinare sulla CCS. Con questo progetto, vengono iniettate 1.8 milioni di tonnellate di CO_2 all'anno, raggiungendo, fino a oggi, oltre 35 milioni di tonnellate stoccate.

Supportato da diverse agenzie governative e aziende, il progetto Weyburn ha stabilito un modello di successo per la commercializzazione della CCS, per giunta senza sovvenzioni. Inoltre, ha dimostrato che è possibile catturare e stoccare la CO_2 generata dalla combustione del carbone, un contributo significativo poiché il carbone rappresenta una delle principali fonti di emissioni di carbonio (Ma et al., 2022).

Ad oggi, le strutture di cattura del carbonio negli Stati Uniti catturano circa il 90% della CO_2 rispetto ad altri paesi. L'Europa ha il secondo maggior numero di progetti, seguita dalla regione Asia-Pacifico. Due grandi progetti finanziati dal governo australiano sono attualmente in corso in Malesia e Indonesia, ed il numero di tonnellate metriche di CO_2 catturate dalle strutture CCS è aumentato da 10,3 milioni nel 2020 a 15 milioni nel 2021.

Inoltre, i progetti di cattura, stoccaggio e utilizzo della CO_2 (CCUS) stanno guadagnando popolarità in varie industrie; tra cui la produzione di acciaio, cemento, fertilizzanti e prodotti chimici. Il Summit Carbon Solution Network, il più grande progetto CCS al mondo situato negli Stati Uniti, sta aiutando 31 impianti di bioetanolo a trasportare e immagazzinare CO_2, con una capacità di 11 milioni di tonnellate metriche nel 2021. Il progetto Aramis nei Paesi Bassi sta catturando una quantità significativa di CO_2, subito dopo il progetto Summit Carbon.

Infine, molti progetti per il deposito dell'anidride carbonica nel terreno hanno preso piede negli ultimi anni: il DF2 della California, Hatfield nel 2014, con 5 milioni di tonnellate all'anno di iniezione di CO_2; Mongstad nel 2014, con 1,5 milioni di tonnellate all'anno di iniezione di CO_2; Greenger in Cina nel 2015, con 0,7 milioni di tonnellate all'anno di iniezione di anidride carbonica; e Trailblazer negli Stati Uniti, costruito nel 2014 con 4,3 milioni di tonnellate all'anno di iniezione di CO2. Anche il progetto Genesse in Canada, costruito nel 2015, con una capacità di 3,6 milioni di tonnellate di CO2 all'anno (Baskaran et al., 2024).

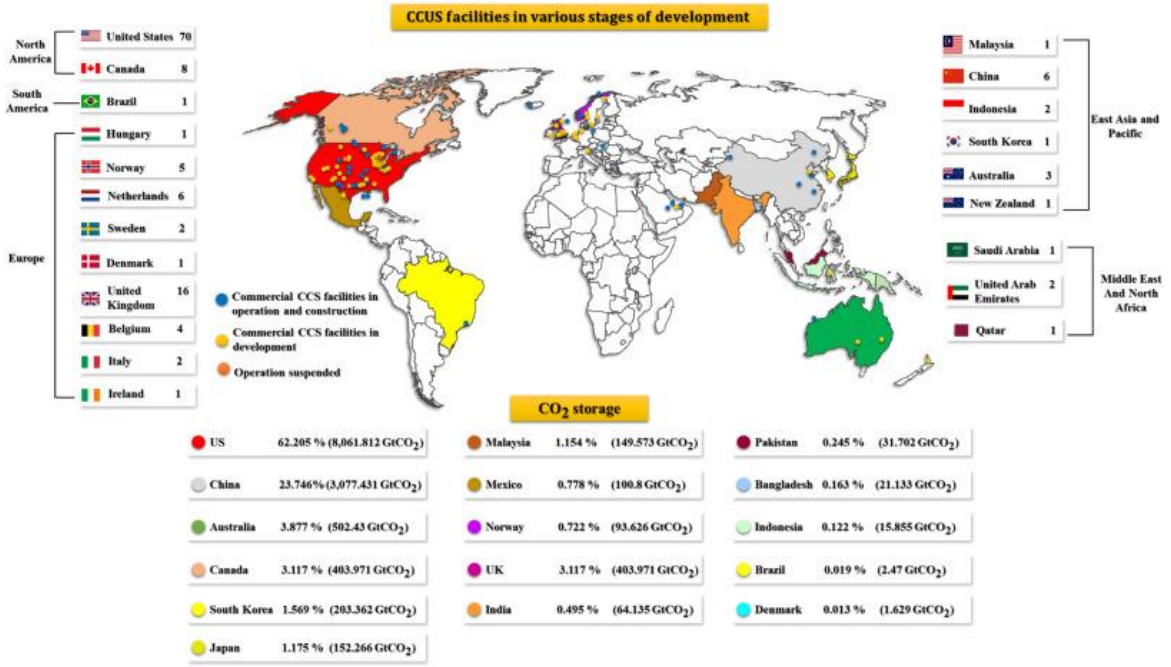

Figura 1: impianti CCS nel mondo (Davoodi et al., 2023)

2.1 Stato dell'Arte

Il processo di CCS avviene in diverse fasi, con il primo passo che consiste nella rimozione della CO_2, attraverso diverse tecniche specifiche, che verranno analizzate in questo capitolo. Una volta catturata e compressa in forma liquida, per facilitarne il trasporto, la CO_2 viene immagazzinata attraverso l'iniezione in rocce porose nel sottosuolo o sotto i fondali oceanici, con un'azione nota come geo-sequestro. Questo processo sfrutta le rocce porose per trattenere la CO_2 iniettata, similmente allo stoccaggio del petrolio e del gas nel sottosuolo. La CO_2 iniettata riempie i pori all'interno delle rocce, e viene trattenuta da uno strato impermeabile che la ricopre. Sia i bacini onshore che offshore possono essere utilizzati per lo stoccaggio geologico della CO_2 (Podder et al., 2023).

D’altro canto, la CO_2, specialmente se con un alto grado di purezza, può essere riutilizzata per una grande varietà di scopi, come il miglioramento del recupero del petrolio, la produzione di carburanti sintetici, o idrocarburi, e l'uso nella lavorazione di alimenti e bevande. I gas di scarico combusti provenienti dagli impianti industriali possono, quindi, diventare una preziosa risorsa per questi processi.

La scelta della tecnologia di Carbon Capture da adottare deve considerare diversi fattori critici per garantire un'efficace riduzione delle emissioni, e quindi una riduzione del potenziale di riscaldamento globale.

Uno dei principali fattori da considerare è il volume richiesto per la realizzazione dell’impianto di cattura: questo è particolarmente rilevante nelle applicazioni marittime, dove lo spazio è limitato. Le navi, infatti, spesso non hanno ampi spazi disponibili, e sono quindi necessarie strutture compatte e adattabili agli spazi ristretti presenti a bordo (Risso et al., 2023).

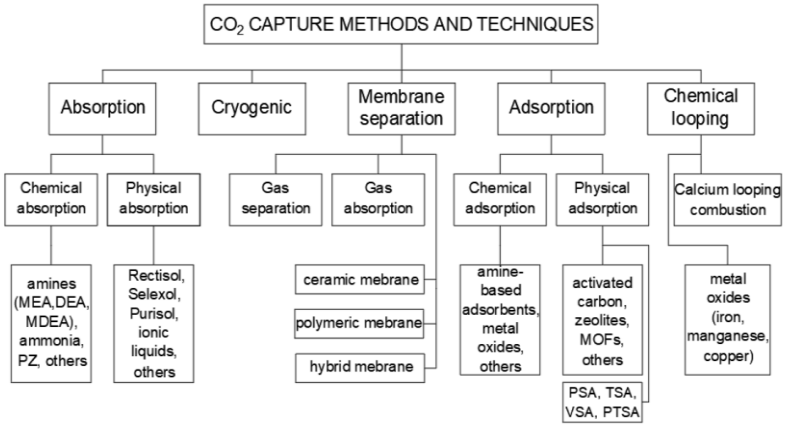

Figura 2: tecniche di Carbon Capture (Madejski et al., 2022)

Anche la gestione e la logistica del caricamento e scaricamento dei materiali devono essere attentamente progettate, per garantire un funzionamento sicuro ed efficiente. Le tecnologie di cattura della CO_2 spesso richiedono l'uso di intermedi chimici, che devono essere prodotti, rigenerati e stoccati in condizioni di sicurezza.

Nei processi di assorbimento, ad esempio, l'uso di solventi richiede speciali recipienti di stoccaggio che garantiscano la sicurezza durante il trasporto e l'utilizzo. Analogamente, nelle celle a combustibile a carbonato fuso, l'uso di idrogeno richiede una gestione attenta del carburante a bordo, con misure di sicurezza che devono essere rigorosamente rispettate per prevenire rischi di esplosioni o incendi.

Inoltre, i sistemi CCS richiedono molta alimentazione per operare: è perciò necessaria un'accurata analisi energetica per valutare l'impatto complessivo dell'impianto. Se un sistema fosse troppo energivoro, l'aumento del consumo potrebbe essere compensato dalla necessità di bruciare più combustibile, annullando così parte dei benefici della cattura della CO_2. È quindi essenziale trovare un equilibrio tra l'energia necessaria per il funzionamento del sistema e l'effettiva riduzione delle emissioni di CO_2.

Le principali tecnologie ad oggi disponibili ed utilizzate si dividono in processi pre-combustione e post-combustione. Maggiore enfasi sarà data all'analisi delle tecniche post-combustione, che utilizzano solventi come, ad esempio, i liquidi ionici, sui quali è incentrata questa trattazione.

2.2 Tecniche Pre-Combustione

La cattura del carbonio pre-combustione è un processo in cui il combustibile (carbone, gas naturale o biomassa) viene convertito in una miscela di monossido di carbonio (CO) e idrogeno (H_2), tramite processi di gassificazione o reforming.

In particolare, il combustibile viene trattato in un gassificatore con ossigeno e vapore, producendo syngas, una miscela di CO e H_2. Successivamente, attraverso la reazione water-gas shift, il CO viene convertito in CO_2 e H_2, permettendo la separazione della CO_2 prima della combustione finale. Questa CO_2 viene poi catturata e stoccata o utilizzata, mentre l'H_2 viene impiegato come combustibile nelle turbine a gas (Madejski et al., 2022). Questa tecnica permette una separazione efficace della CO_2 grazie all'alta concentrazione del gas nel syngas prima della combustione. Tuttavia, il processo richiede importanti unità di gassificazione e comporta, quindi, elevati costi energetici.

Durante il processo, il syngas viene filtrato per rimuovere ceneri e impurità, e la CO_2 viene catturata in un separatore dedicato (Podder et al., 2023). I metodi di cattura includono sia assorbenti chimici, come i carbonati, sia solventi fisici, come il glicole polipropilenico e il metanolo, che vengono utilizzati commercialmente per questi scopi. Un metodo alternativo, il ciclo del calcio, utilizza CaO per assorbire la CO_2 e successivamente $CaCO_3$ per rilasciare CO_2 a temperature ottimali, riducendo il consumo energetico grazie all'uso del calore di scarto del gassificatore(Madejski et al., 2022). Un esempio concreto di successo è rappresentato dall'impianto di Port Arthur negli Stati Uniti, che ha catturato con successo un milione di tonnellate di CO_2 utilizzando la tecnologia di adsorbimento a doppia pressione, dimostrando un'alta efficienza di cattura e purificazione dell'idrogeno. (Baskaran et al., 2024).

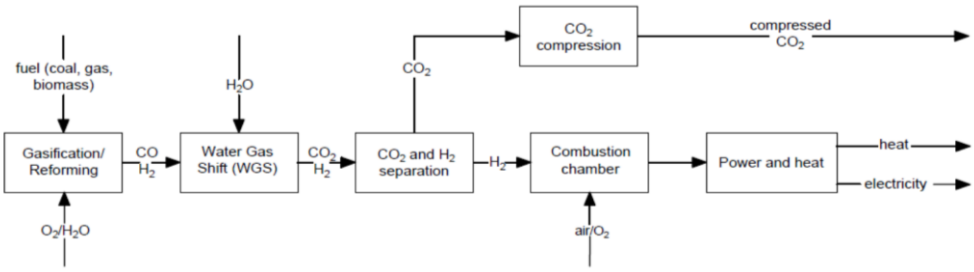

Figura 3: processo Pre-Combustione (Madejski et al., 2022)

2.3 Tecniche Post-Combustione

Il processo standard di cattura del carbonio post-combustione (Post Combustion Capture, PCC) è formato da una serie di componenti chiave: assorbitore, pompe, scambiatori di calore, stripper e compressore. I gas di scarico, dopo la combustione dei combustibili fossili, vengono inviati all'unità di assorbimento, dove circa l'80-90% della CO_2 viene catturata grazie all'azione di un solvente, come, ad esempio, la monoetanolammina (MEA) o le miscele di ammine e piperazina. L'assorbitore è progettato per migliorare il trasferimento di massa tra i gas di scarico e il solvente, utilizzando pacchi strutturati.

La soluzione, ora ricca di anidride carbonica, viene riscaldata attraverso uno scambiatore di calore e inviata alla colonna di stripper, dove avviene la separazione della CO_2 catturata. In questa fase, il solvente viene rigenerato e riciclato nell'assorbitore come soluzione povera, mentre la CO_2 separata viene compressa e, successivamente, stoccata.

L'efficienza del processo PCC dipende dalle reazioni chimiche che avvengono nelle colonne di assorbimento e stripper. Le reazioni di assorbimento avvengono attraverso meccanismi di reazione acido-base, in cui ciascun solvente reagisce con la CO_2 formando bicarbonato e carbamato, migliorando così le prestazioni di assorbimento e rigenerazione (Alalaiwat & Khan, 2024).

La tecnologia PCC deve affrontare diverse sfide, tra cui la gestione dei gas di scarico con una bassa concentrazione di CO_2 e l'alto costo delle attrezzature necessarie per raggiungere una purezza della CO_2 superiore al 95,5%. Prima della separazione della CO_2, infatti, i gas di scarico devono passare attraverso processi di denitrificazione, desolforazione e rimozione della polvere, per eliminare N2, NOx e SO2. Dopo la separazione, invece, la CO_2 deve essere compressa in fluidi supercritici e stoccata in serbatoi geologici.

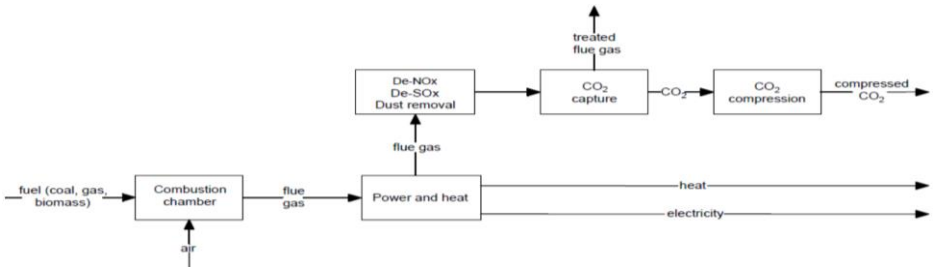

Figura 4: processo Post-Combustione (Madejski et al., 2022)

I processi PCC più comuni includono l'assorbimento chimico con ammine, l'adsorbimento, la separazione a membrana e il looping chimico. Sebbene l'assorbimento con ammine sia efficace, presenta svantaggi come la degradazione del solvente, la perdita per evaporazione, la corrosività e alti costi operativi. Per migliorare l'efficienza e ridurre i costi, si stanno sviluppando tecnologie alternative come i framework metallo-organici (Metal-Organic Frameworks, MOF) e membrane avanzate. I MOF, ad esempio, offrono un'elevata efficienza nella separazione della CO_2 grazie alla loro struttura porosa personalizzata e alla morfologia, anche se sono limitati dall'instabilità strutturale e dall'intenso fabbisogno energetico (Baskaran et al., 2024).

Nonostante il metodo di pre-combustione offra maggiore efficienza e minori costi operativi, il PCC rimane la tecnologia più matura e ampiamente utilizzata per la cattura e lo stoccaggio del carbonio. Si prevede che la commercializzazione di solventi a base di ammine per centrali a carbone sarà disponibile entro i prossimi anni, con continui miglioramenti nell'efficienza e nella riduzione dei costi (Podder et al., 2023).

2.3.1 Adsorbimento

Il processo PCC basato sull'adsorbimento prevede il trasferimento di molecole di CO_2 presenti nei gas di scarico, a pressione atmosferica e ad alta temperatura, sul sito attivo di un materiale solido, dove esso ha una forte affinità superficiale. Le molecole di CO_2 attaccate, chiamate adsorbato, formano una pellicola sulla superficie del materiale solido, denominato adsorbente. Ciò avviene secondo due tipi di adsorbimento: fisico e chimico, analizzati in seguito (Akinola et al., 2022). Una volta terminato il processo di adsorbimento, vi è la necessità di rilasciare le molecole di CO_2, rigenerando l'adsorbente. Questo processo, chiamato desorbimento, può essere effettuato utilizzando diverse tecniche, tra cui il riscaldamento diretto, con gas caldi tra 150 e 200°C, o indiretto, con gli scambiatori di calore. Una delle principali problematiche nel desorbimento è il consumo energetico elevato e la necessità di mantenere la stabilità termica e chimica degli adsorbenti nel tempo. Studi recenti stanno esplorando modi per migliorare l'efficienza del desorbimento e ridurre i costi energetici associati (Baskaran et al., 2024).

L'adsorbimento fisico coinvolge interazioni deboli, principalmente forze di Van der Waals ed elettrostatiche, senza la formazione di nuovi legami chimici; ciò rende l'energia richiesta per l'adsorbimento e il successivo desorbimento dell'adsorbente relativamente bassa (Petrovic et al., 2022). Questa reazione avviene grazie alle porosità degli adsorbenti e, più in generale, alle loro caratteristiche fisiche, ed è fortemente correlata anche alle proprietà dei gas: tra le più importanti troviamo la forma, il punto di ebollizione, il diametro cinetico, la polarizzabilità e il momento di dipolo. La CO_2, ad esempio, ha una forma lineare e un diametro cinetico piccolo, che le consentono di entrare facilmente nei micropori confinati di un adsorbente con sito attivo.

Quanto ai materiali utilizzati come adsorbenti, invece, essi includono elementi carboniosi come il carbone attivo, i nanotubi di carbonio e il grafene, nonché materiali non carboniosi come le zeoliti. Il carbone attivo, ad esempio, è ottimo grazie alla sua alta stabilità, resistenza al calore e ai prodotti chimici, basso costo di sintesi e buona area superficiale e volume dei pori. I nanotubi di carbonio, invece, offrono una buona resistenza termica, chimica e, di conseguenza, un'ottima capacità di adsorbimento, che può essere ulteriormente migliorata con l'impregnazione di ZnO. Il grafene è efficace specialmente quando esposto alla luce solare o alla radiazione ultravioletta durante la sintesi (Baskaran et al., 2024). Infine, le zeoliti, mostrano buone capacità di adsorbimento della CO2 grazie alla loro struttura microporosa.

L'adsorbimento chimico, invece, è caratterizzato dalla formazione di legami con un'energia superiore a circa 0.5 eV per molecola, che sfruttano spesso le reazioni acido-base di Lewis, generando interazioni molto più forti rispetto all'adsorbimento fisico.

In questo processo, la reazione avviene in zone specifiche sulla superficie dell'adsorbente: ad esempio, i materiali carboniosi possono essere trattati per introdurre eteroatomi come l'azoto, che crea siti attivi capaci di interagire chimicamente con la CO_2. I gruppi funzionali contenenti azoto, infatti, in particolare ad alte temperature (120°C circa), conferiscono carattere basico alla superficie, rendendo questi siti attrattivi per l'anidride carbonica (Petrovic et al., 2022).

2.3.2 Assorbimento

L'assorbimento è un processo che si basa sull'utilizzo di solventi liquidi, che catturano la CO_2 dai gas di scarico, convertendola in una forma che può essere successivamente rilasciata e stoccata. Anch'esso, come l'adsorbimento, può essere classificato in due categorie principali: chimico e fisico.

Nell'assorbimento chimico, analogamente all'adsorbimento, il solvente reagisce con la CO_2 attraverso reazioni di neutralizzazione acido-base. Ad esempio, le ammine, come la MEA, sono comunemente utilizzate per la loro alta capacità di assorbimento della CO_2 e per la loro efficienza nel catturare fino al 90% della CO_2 presente nei gas di scarico.

La MEA, tuttavia, presenta alcuni svantaggi, tra cui l'alto consumo energetico per la rigenerazione del solvente, la degradazione del solvente stesso a causa della presenza di SO_2 e NO_2 nei gas di scarico, e la corrosione delle attrezzature dovuta alla presenza di O_2 (Hong, 2022).

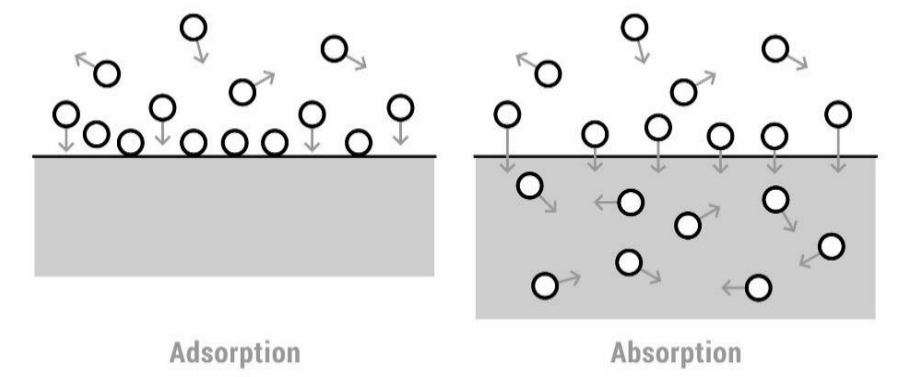

Figura 5: differenza visiva tra adsorbimento e assorbimento (Absorption or Adsorption? | Durpro)

Per affrontare questi problemi, sono stati sviluppati altri solventi chimici come la dietanolammina (DEA), la metildietanoloammina (MDEA), l'ammoniaca liquida e la piperazina. Questi solventi offrono vantaggi quali minore corrosività, maggiore stabilità termica e minore consumo energetico per la rigenerazione. Ad esempio, la combinazione di MDEA e piperazina riduce significativamente il fabbisogno energetico rispetto alla sola MEA. Inoltre, anche il carbonato di potassio acquoso ($K2CO_3$) viene spesso usato come assorbente chimico, è noto per la sua bassa tossicità e facile rigenerazione. La velocità di assorbimento del $K2CO_3$, generalmente bassa, può essere migliorata con l'aggiunta di promotori come la DEA e la PZ, aumentando l'efficacia del processo (Baskaran et al., 2024).

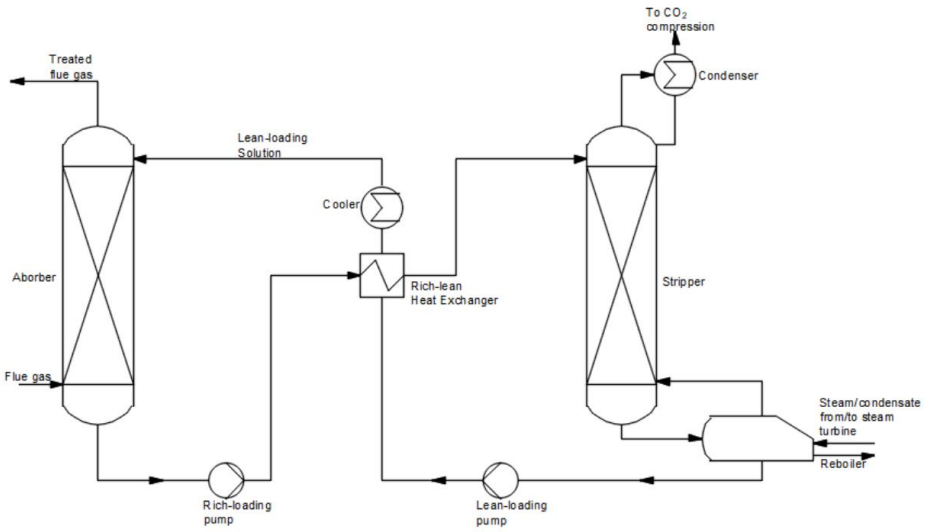

Figura 6: impianto PCC con assorbimento chimico (Madejski et al., 2022)

L'assorbimento fisico, invece, si basa sulla solubilità della CO_2 in solventi organici e variazioni di temperatura e pressione. Solventi fisici comuni includono il dimetil-etere del polietilenglicole (DEPG), il metanolo refrigerato e il N-metil-pirrolidone (NMP). Questi solventi sono particolarmente efficaci a pressioni parziali elevate della CO_2, rendendoli adatti anche per la cattura pre-combustione.

2.3.3 Liquidi Ionici

Un’altra tecnologia emergente nel campo dell’assorbimento consiste nell'uso dei liquidi ionici: sali che, a temperatura ambiente o poco sopra, si presentano allo stato liquido, e che offrono vantaggi significativi in termini di bassa volatilità, buona stabilità termica e bassi requisiti energetici per la rigenerazione. Essi beneficiano, inoltre, di un’alta solubilità della CO_2 e della capacità di personalizzazione delle proprietà chimiche.

Questi materiali possono essere divisi in due categorie principali: i liquidi ionici protici (PILs), che possono donare un protone, e i liquidi ionici aprotici (AILs), che non possono farlo (Faisal Elmobarak et al., 2023).

Le prime ricerche hanno evidenziato che l'interazione tra la parte anionica dei liquidi ionici e la CO_2 gioca un ruolo cruciale nella capacità di assorbimento, superando in prestazioni i solventi tradizionali. Ad esempio, il liquido ionico standard [emim][Tf2N] ha mostrato un'elevata capacità di assorbimento della CO_2 in diverse condizioni operative, mentre i solventi molecolari offrono buone prestazioni solo a pressioni medio-alte. La capacità di assorbimento di vari PILs basati su anioni contenenti gruppi funzionali diversi è stata testata, e si è scoperto che questi liquidi ionici mostrano un'elevata capacità di assorbimento grazie alla forte interazione elettrostatica e ai legami a idrogeno presenti (Faisal Elmobarak et al., 2023).

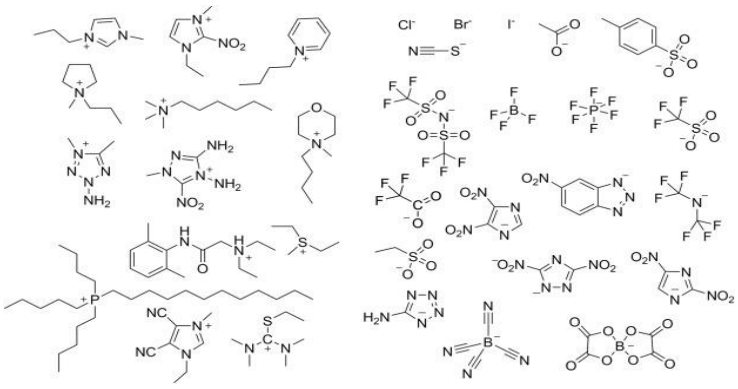

Figura 7: principali strutture Ioniche liquide (Gaijon et al., 2022)

I IL, non solo catturano efficacemente la CO_2, ma possono anche facilitare la sua conversione in prodotti utili attraverso approcci di riduzione chimica, biochimica, fotochimica, termochimica ed elettrochimica. Ad esempio, i liquidi ionici possono essere utilizzati per convertire la CO_2 in carbonati ciclici attraverso reazioni di ciclo-addizione; mentre nei processi elettrochimici, essi possono migliorare l'efficienza e la capacità di conversione della CO_2, sfruttando l'energia elettrica proveniente da fonti rinnovabili come l'energia solare (Podder et al., 2023).

La capacità di assorbimento della CO_2 dei liquidi ionici varia notevolmente, con una media di 6.03 mol CO_2/kg per l'assorbimento chimico tramite liquidi ionici convenzionali, fino a 8.84 mol CO_2 /kg per liquidi ionici specifici per compiti funzionalizzati con ammine (Hong, 2022). Tuttavia, l'alto costo dei liquidi ionici e il loro rapporto prezzo/prestazioni non ancora competitivo rispetto ai solventi commerciali esistenti, rappresentano una sfida significativa per la loro applicazione su larga scala.

Per cercare di risolvere questi problemi, e, oltretutto, per personalizzare al meglio ogni tipo di applicazione, una soluzione attuale risiede sicuramente nella ricerca e nella progettazione di liquidi ionici alternativi, tramite tecniche come il computer-aided molecular design. Per far questo, è necessario predire e valutare la capacità di assorbimento dei IL in diverse condizioni operative di temperatura e pressione, basandosi sulla loro struttura molecolare. La tesi in questione ha proprio questo obiettivo, riuscire a fornire un algoritmo in grado di stimare efficacemente la solubilità della CO_2 in vari liquidi ionici, conoscendo anche il modello matematico con cui viene svolta la regressione.

3 Algoritmi di Intelligenza Artificiale per la modellazione dei processi di Carbon Capture

3.1 Applicazioni industriali

L'uso del Machine Learning e, più in generale, dell'Intelligenza Artificiale, sta rivoluzionando i processi di Industria 4.0 attraverso numerose applicazioni, migliorando l'efficienza, la qualità e l'adattabilità dei sistemi di produzione. Gli algoritmi di AI, infatti, possono gestire, elaborare e scambiare informazioni, riuscendo ad eseguire operazioni con un elevato tasso di precisione e nel minor tempo possibile.

Questi nascono da diversi tipi di approcci e aree di ricerca: il ragionamento probabilistico e fuzzy, per la progettazione di programmi che agiscono in condizioni di incertezza riguardo all'ambiente esterno al programma; la pianificazione, per determinare automaticamente una sequenza di azioni per raggiungere un obiettivo; l'apprendimento automatico, per la realizzazione di programmi che migliorano le loro prestazioni in base all'esperienza; le reti neurali, per applicazioni basate su complesse funzioni non lineari apprese automaticamente dai dati; e gli algoritmi genetici, come il Grey Wolf Optimizer (GWO), di cui si parlerà in seguito.

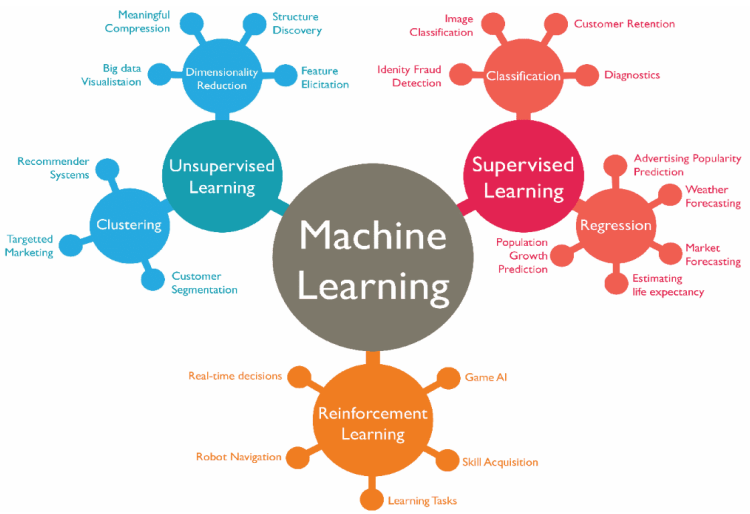

Figura 8: tecniche di ML (10 Companies Using Machine Learning in Cool Ways)

Alcuni tra gli algoritmi più conosciuti sono la Support Vector Machine (SVM), il K-Nearest Neighbor (KNN), il Decision Tree (DT) e la Random Forest (RF). A questi, si aggiungono poi i diversi metodi di deep learning (DL), che sono in grado di identificare relazioni complesse nei dati e di migliorare le prestazioni dei modelli: alcuni esempi sono la rete neurale ricorrente (Recurrent Neural Network, RNN), la rete neurale convoluzionale (Convolutional Neural Network, CNN) e la Rete Neurale con Memoria a Lungo Termine (Long Short-Therm Memory, LSTM); queste sono specificamente utili per grandi dataset, serie temporali e analisi di immagini.

Una delle principali applicazioni del ML in ambito industriale è sicuramente l'ottimizzazione della Supply Chain (SC), che migliora la previsione della domanda, il controllo dei magazzini e la logistica. Ad esempio, la Time Series Analysis (TSA) e il Natural Language Processing (NLP) vengono utilizzati per analizzare i comportamenti e le preferenze dei clienti, mentre i sistemi di visione artificiale basati sul deep learning facilitano il controllo e il rifornimento delle scorte.

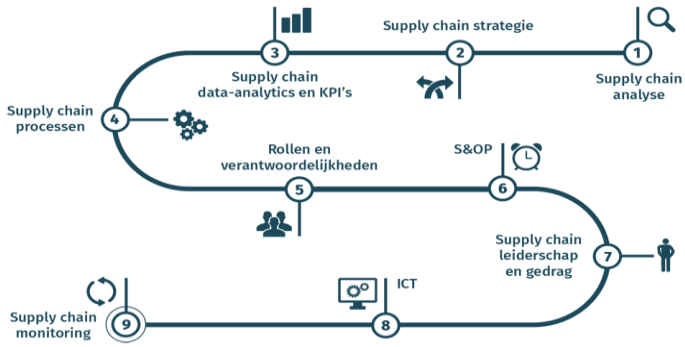

Figura 9: ottimizzazione della supply chain (A Guide to Digital Twin Development - Visartech Blog)

L'ispezione e il monitoraggio dei processi tramite sistemi di visione artificiale rappresentano un'altra applicazione critica. Sensori economici, come le fotocamere RGB, combinati con algoritmi di ML, possono eseguire controlli delle parti ad alta velocità e monitorare continuamente i processi di produzione, migliorando la qualità dei prodotti.

Inoltre, anche per quanto riguarda la previsione del consumo energetico, modelli come le reti neurali profonde possono tornare molto utili per riuscire a conoscere meglio la tendenza dei consumi che un'azienda dovrà affrontare, permettendo un'ammortizzazione dei costi ed evitando ritardi nella produzione.

Una delle innovazioni più importanti di Industria 4.0 è rappresentata sicuramente dai Digital Twin (DT), che offrono numerosi vantaggi in termini di diagnosi, valutazione e previsione dei processi produttivi. Un DT è una replica virtuale di un prodotto, processo o sistema, che utilizza dati e impostazioni predefinite per simulare le prestazioni e il comportamento del corrispettivo reale. Questa tecnologia consente alle aziende di analizzare e ottimizzare i processi produttivi in tempo reale.

I DT possono monitorare continuamente lo stato delle macchine e delle linee di produzione, rilevando immediatamente anomalie o deviazioni dai parametri ottimali. Infatti, è importante precisare che una delle caratteristiche principali di questa tecnologia è la connessione real-time: i Digital Twin si aggiornano in tempo reale in base agli input che arrivano dal sistema reale. Quindi quando si verifica un problema, si riesce ad individuarne rapidamente la causa (7 Applications of Machine Learning in Manufacturing in 2023).

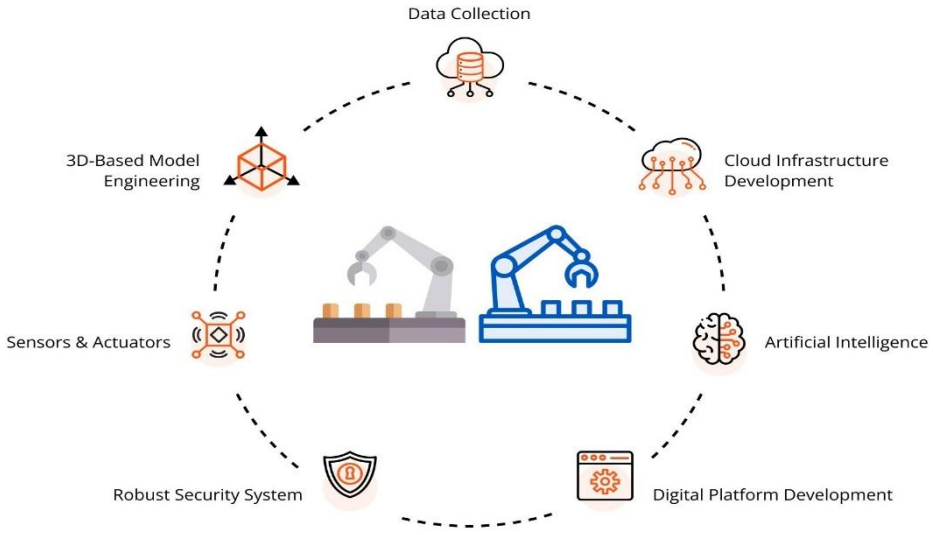

Figura 10: elementi principali di un Digital Twin (A Guide to Digital Twin Development - Visartech Blog)

Essi consentono di eseguire simulazioni dettagliate dei processi produttivi, individuando inefficienze e colli di bottiglia, ottimizzando la produzione e non interrompere le operazioni. Attraverso l'analisi dei dati storici e contemporanei, invece, si riesce a prevedere il comportamento futuro dei processi produttivi, consentendo una gestione proattiva e l'implementazione di strategie di miglioramento continuo. Infine, i DT permettono di sperimentare con nuovi design in un ambiente virtuale prima di passare alla produzione reale, riducendo i costi e i tempi di sviluppo dei nuovi prodotti, consentendo modifiche basate sui feedback. Le linee di produzione possono essere adattate e ottimizzate per rispondere rapidamente alle esigenze del mercato e alle richieste specifiche dei clienti, aumentando la flessibilità e la capacità di soddisfare le esigenze personalizzate senza compromettere l'efficienza.

3.2 Stato dell'Arte

Anche per quanto riguarda i processi di Carbon Capture, una modellazione basata su algoritmi di AI, offre la possibilità di portare a un livello più avanzato questo settore, consentendo di migliorarne l'efficienza, il controllo e la qualità.

Le tecniche di ML sono già state ampiamente adottate per affrontare una serie di problemi nel settore CCs, producendo anche risultati molto positivi. Tra le applicazioni più comuni troviamo sicuramente la simulazione dei processi di cattura pre-combustione e post-combustione, la predizione della solubilità della CO_2 e la progettazione di nuovi materiali per l'adsorbimento e l'assorbimento, come i MOF e i IL. Per quanto concerne il settore dello stoccaggio, invece, sono molto importanti la previsione dei vari parametri fisici e l'analisi di stabilità per i modelli di serbatoio, nonché il supporto decisionale (Davoodi et al., 2023).

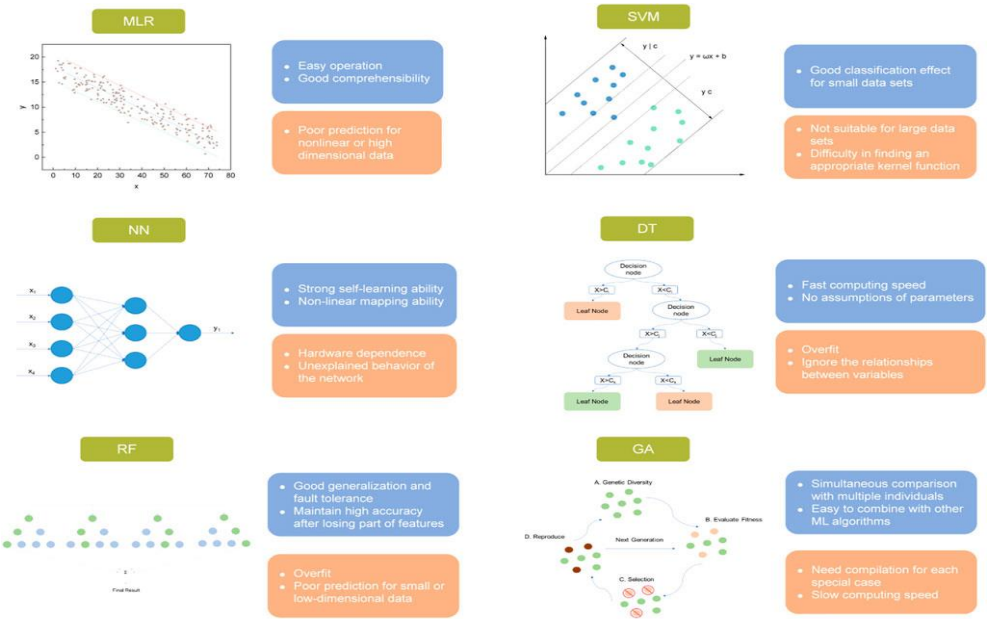

Figura 31: vantaggi e svantaggi di varie tecniche di ML nella separazione del Carbonio (Yang et al., 2023a)

Nell’ambito della progettazione di nuovi materiali, come accennato pocanzi, di notevole importanza rimangono sicuramente lo studio e l’ottimizzazione dei MOF, vista la loro versatilità e le loro prestazioni nel sequestro della CO_2.

L'uso del machine learning nella cattura assistita dai MOF risale al 2014, quando fu adottato per riconoscere i modelli di adsorbimento della CO_2 su oltre 32.000 frameworks diversi. Uno dei primi approcci utilizzò il modello SVM per identificare i MOF più performanti, scoprendo che una distanza interatomica tra 6 e 9 Å era ottimale per intrappolare la CO_2.

Nel 2018, le proprietà chimiche dei MOF furono inserite negli studi grazie a metodi topologici. Le Gradient Boosted Machine e le Reti Neurali si dimostrarono i modelli più accurati per correlare questi descrittori con le prestazioni dei MOF (Yang et al., 2023b).

Nonostante l'inclusione delle caratteristiche chimiche, le proprietà fisiche, come dimensione dei pori, frazione vuota, area superficiale e densità, restarono i fattori predominanti per determinare la capacità di cattura della CO_2.

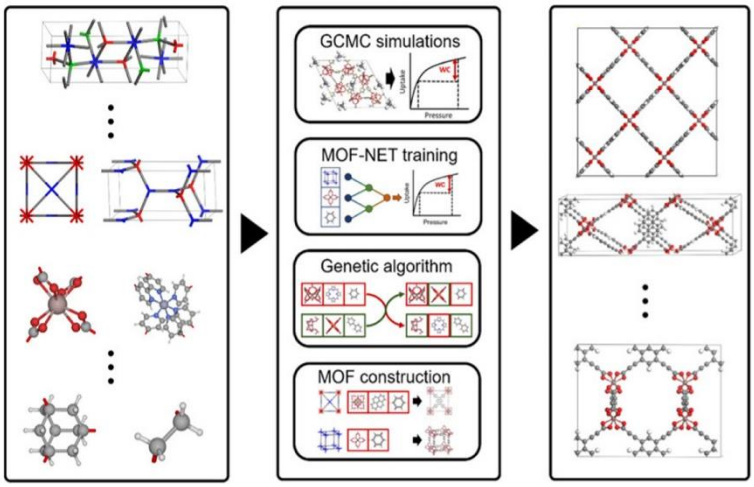

Figura 14: progettazione di MOF con tecniche di ML (Yu Tsivadze et al., 2023)

D’altro canto, altri studi recenti portati avanti da Hosseinpour et al., hanno esteso l'analisi ML per studiare il comportamento di altri componenti gassosi, come CH_4 e N_2, studiando la selettività dei MOF rispetto ad essi.

Utilizzando modelli di regressione lineare e reti neurali, è stato scoperto che, anche in questo caso, una combinazione ottimale di area superficiale, frazione vuota e dimensione dei pori è essenziale per massimizzare la selettività.

Tuttavia, contrariamente a ciò che valeva per la CO_2, in questo studio si evince l'importanza di includere all'interno del modello anche le proprietà chimiche dei MOF, come il momento dipolare e la distribuzione della carica, al pari di quelle fisiche, per migliorare ulteriormente l'accuratezza delle ricerche (Hosseinpour et al., 2023).

Con l'inclusione dei parametri fisici e chimici, gli studi iniziarono ad analizzare le prestazioni dei MOF in scenari pratici. Nel 2021, un modello basato su una rete neurale esaminò la separazione della CO_2 in presenza di vapore acqueo, suggerendo che i MOF con area superficiale e diametro dei pori limitati mostravano prestazioni ottimali.

Ricerche recenti hanno analizzato anche le prestazioni delle zeoliti, tramite tecniche di machine learning: grazie a modelli come l'Hybrid-ANFIS (Adaptive Neuro-Fuzzy Inference System), è stato possibile stimare l'adsorbimento della CO_2 con un alto grado di precisione, basandosi su parametri come l'incremento di pressione e la temperatura.

I risultati hanno mostrato che l'incremento di pressione ha un effetto maggiore sull'adsorbimento rispetto alla temperatura, eccetto per alcune zeoliti che mostrano un'evidente sensibilità al calore (Priya et al., 2023).

Negli ultimi anni, l'applicazione del ML ha aperto nuove prospettive anche per quanto riguarda il mondo dei IL, cercando di ovviare alle complessità dei tradizionali Metodi di Progettazione Assistita da Computer (CAMD). Questi, infatti, sono basati su modelli termodinamici che richiedono calcoli intensivi, e spesso non valutano adeguatamente la diversità di questi composti ionici.

Uno dei primi studi significativi è stato condotto da Sun et al: sono stati sviluppati algoritmi di ML per prevedere le proprietà fisiche e chimiche degli ILs, utilizzando dati sperimentali e relazioni termodinamiche come input.

I modelli hanno dimostrato una capacità predittiva superiore rispetto ai metodi tradizionali, una maggiore rapidità di screening e una minore deviazione rispetto ai dati sperimentali. Ad esempio, è stata creata una Rete Neurale Feed-Forward (FF-ANN) per prevedere la viscosità degli ILs, che mostrava una maggiore accuratezza rispetto ai modelli esistenti (Sun et al., 2023a).

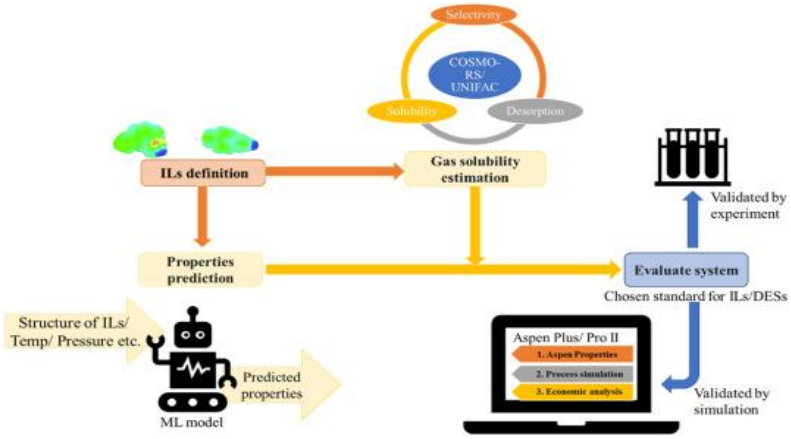

Figura 15: progettazione di IL con tecniche di ML (Sun et al., 2023b)

Durante l'ultimo anno, Kuroki et al. hanno analizzato 6991 composti ionici, utilizzando delle caratteristiche elettrochimiche e le curve di energia potenziale di interazione come input dell'algoritmo di modellazione. Successivamente è stata effettuata una ricerca, sempre assistita dal ML, per identificare la migliore combinazione catione/anione, selezionando il [P66614][PFOS] tra 402114 ioni (Kuroki et al., 2023). Questo studio ha evidenziato come il ML possa essere utilizzato non solo per prevedere singole proprietà, bensì per ottimizzare l'intero processo di cattura della CO_2, tenendo conto di più variabili contemporaneamente.

Inoltre, in un altro progetto è stato evidenziato come, attraverso tecniche di ML e DL, si riescano a identificare rapidamente combinazioni ottimali di cationi e anioni, riducendo significativamente il tempo necessario per la sperimentazione e l'analisi dei dati. L'utilizzo di questi algoritmi ha permesso di migliorare la comprensione delle relazioni struttura-proprietà dei IL, facilitando la progettazione di nuovi composti, con caratteristiche su misura per specifiche applicazioni (Yang et al., 2023b).

In quest'analisi si evince per di più che la modellazione basata sul ML può essere integrata con metodi termodinamici tradizionali, per migliorare ulteriormente le prestazioni e l'efficienza dei processi in condizioni operative reali.

Recentemente, alcuni algoritmi di AI sono stati usati anche per prevedere la solubilità della CO_2 in soluzioni saline. In particolare, in uno studio, il DT e il RF sono stati presi come ottimizzatori per una ANN con un solo hidden layer da 5 neuroni, con funzione di attivazione ReLU e e il metodo di discesa del gradiente stocastico con backpropagation per minimizzare l'errore e addestrare la rete. Sono stati considerati dati di solubilità generati numericamente con la relazione di Setschenow; come input, invece, le proprietà degli ioni (carica ed energia libera di idratazione), le loro concentrazioni molari, la pressione e la temperatura, sono risultate le grandezze più adatte (Ratnakar et al., 2023).

Infine, si introduce lo studio realizzato da Song et al. nel 2020, sul quale ci si è basati per questa trattazione. Lo studio raccoglie un totale di 10116 valori di solubilità della CO_2 in 124 diversi IL, con i cationi che includono imidazolium, pirrolidinio, piridinio, piperidinio, ammonio, fosfonio e solfonio, mentre gli anioni comprendono tetrafluoroborato [BF_4], cloruro [Cl], dicianammide [DCA], nitrato [NO_3], esafluorofosfato [PF_6], tiocianato [SCN], tricianometanide [$C(CN)_3$], solfato acido [HSO_4], bis(trifluorometilsolfonil)ammide [Tf_2N], metilsolfato [$MeSO_4$], e altri.

Nell'articolo in questione sono proposti due modelli di regressione, uno basato su una ANN e l'altro su una SVM. Vengono esposte le differenze sostanziali tra gli approcci, sottolineandone vantaggi e svantaggi. Per l'approccio con la ANN è stata utilizzata una rete a tre strati, dove quello di input riceve informazioni sulla struttura degli IL, la temperatura e la pressione. L'Hidden Layer contiene sette neuroni, e i parametri sono ottimizzati minimizzando la somma degli errori assoluti, utilizzando l'algoritmo di Levenberg-Marquardt. Gli indicatori statistici per il set di test mostrano un Mean Absolute Error (MAE) di 0,0202, e un coefficiente di determinazione (R^2) di 0,9836, dimostrando l'alta qualità delle previsioni.

L'algoritmo SVM, invece, ha una funzione kernel radiale gaussiana, con parametri ottimizzati durante il processo di addestramento. In questo caso si ottiene un MAE di 0,0240 e un R^2 di 0,9783, evidenziando anche in questo caso delle ottime prestazioni (Song et al., 2020).Entrambi i modelli sono in grado di fornire previsioni affidabili in tempi ragionevoli, consentendo una progettazione più snella, con una migliore scelta dei materiali. Di contro però, rimane importante analizzare e stimare i dati con un modello fisico-matematico che rispecchi la reale natura del problema. Gli algoritmi di ANN e SVM, infatti, non derivano dai principi termodinamici e non garantiscono teoreticamente le previsioni degli effetti di temperatura e pressione. Inoltre, attualmente non sono disponibili nei simulatori di processo, rendendo difficile il loro uso diretto per l'ottimizzazione.

4 Materiali e Metodi

Nell’ambito dell’Intelligenza Artificiale e del Machine Learning, gli algoritmi di ottimizzazione giocano un ruolo cruciale nel migliorare le performance dei modelli e nell'ottimizzare l'efficienza computazionale. Questo capitolo si concentra su una dettagliata illustrazione di alcuni dei principali approcci di ottimizzazione, con un'attenzione particolare agli algoritmi di Swarm Intelligence (SI). In particolare, verranno analizzati i principali algoritmi di SI, con un focus approfondito sul Grey Wolf Optimizer (GWO).

L'algoritmo Grey Wolf Optimizer è un potente strumento per analizzare i processi essenziali di elaborazione dei dati, applicabile allo studio di un dataset relativo ai processi di cattura del carbonio.

Inoltre, verrà fornita una guida pratica all'implementazione di questi algoritmi, spiegando in modo dettagliato i passaggi necessari per integrarli efficacemente nei progetti di ricerca e sviluppo. Di seguito vengono introdotti brevemente i passaggi seguiti per le lavorazioni.

Inizialmente è stato studiato il problema in questione, acquisendo dalla letteratura i principi che regolano e influenzano la Solubilità, con particolare focus sulle sue relazioni con le features date. Parallelamente, è stato analizzato il dataset, partendo dalle informazioni contenute nello studio di Song, per poi scendere più nel dettaglio, riguardo la struttura delle varie righe. Conseguentemente, sono state mostrate le distribuzioni dei valori delle features e del target all’interno delle colonne. In aggiunta, sono state graficate le relazioni matematiche tra i valori del dataset, mettendo le basi per la costruzione del modello.

Nella fase successiva, si è cominciato a studiare ed analizzare l’algoritmo GWO, evidenziando quali aspetti sarebbe stato necessario modificare per l’applicazione in questione, e come preparare i dati per facilitarne il funzionamento. Si è continuato, quindi, con azioni di Preprocessing mirate alla semplificazione della struttura dei dati, che verranno analizzate in seguito.

Una volta terminata questa fase, sono state rigorosamente definite le funzioni matematiche, basi di funzionamento dell'algoritmo, che consentono di collegare le features al target, stimandolo con precisione. In concomitanza, è stato personalizzato anche il corpo centrale dell'algoritmo, rendendolo adatto alla trattazione.

Successivamente, tentando di raggiungere i primi risultati, si è deciso di optare per una analisi clusterizzata, frammentando il dataset secondo un criterio preciso (che verrà esposto in seguito), per riuscire a predire la Solubilità con maggior accuratezza. Infine, sono stati plottati i risultati ottenuti, valutando le performance presentate dal modello. Nella figura 16 viene mostrato un diagramma di flusso riassuntivo su quanto detto.

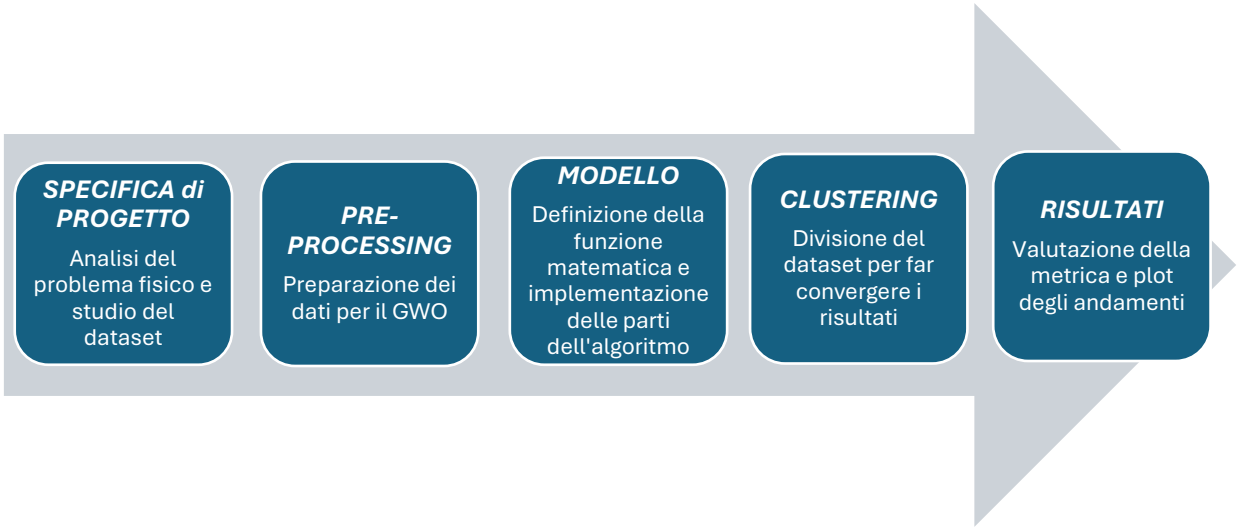

Figura 16: diagramma di flusso delle lavorazioni

4.1 Algoritmi di Swarm Intelligence

Da sempre, in natura, grandi quantità di entità individuali si organizzano spontaneamente in forme diverse, per raggiungere obiettivi funzionali attraverso interazioni locali. Alcuni esempi includono l'auto-assemblaggio molecolare dei flagelli batterici, le decisioni collettive delle colonie di api e i metodi di caccia e di spostamento di vari branchi di animali. Questo fenomeno, chiamato Swarm Intelligence (SI), consente ai gruppi di prendere decisioni superiori rispetto ai singoli membri.

Gli algoritmi di SI, che sono stati utilizzati in questa tesi, nascono come dei metodi computazionali ispirati da questi principi sin dagli studi di Eberhart e Kennedy nel 1995. Essi rappresentano un campo in crescita all'interno del settore dell'AI, combinando computazione evolutiva, reti neurali e sistemi fuzzy.

Questi algoritmi sono spesso utilizzati per fronteggiare problematiche di ottimizzazione, dove i metodi tradizionali falliscono a causa della complessità matematica, o della natura stocastica dello studio (Freitas et al., 2020).

Una caratteristica chiave di queste tecnologie è l'auto-organizzazione: agenti individuali interagiscono tra loro localmente, adattandosi dinamicamente all'ambiente senza necessità di un controllo centralizzato. Infatti, questi algoritmi si distinguono per la decentralizzazione delle decisioni. Pur operando con informazioni parziali, il gruppo è in grado di prendere decisioni collettive superiori, grazie all'integrazione delle conoscenze distribuite. L'approccio biologico di questi algoritmi, inoltre, li rende particolarmente funzionali e versatili; per questo vengono utilizzati in una vasta gamma di applicazioni pratiche, tra cui l'ottimizzazione dei parametri, il riconoscimento di sistemi, e il trattamento di immagini e segnali (Tang et al., 2021). Di seguito si introducono alcuni tra i più importanti algoritmi di SI, delineandone le caratteristiche e il metodo di funzionamento.

Uno dei più conosciuti e studiati algoritmi di SI, come sottolinea la figura 14, è il Particle Swarm Optimization (PSO). Esso fornisce una tecnica di ottimizzazione metaeuristica, riconosciuta per la sua implementazione semplice in problemi complessi e non supervisionati. Si basa su un modello fisico che imita il comportamento collettivo osservato negli stormi di uccelli e nei banchi di pesci.

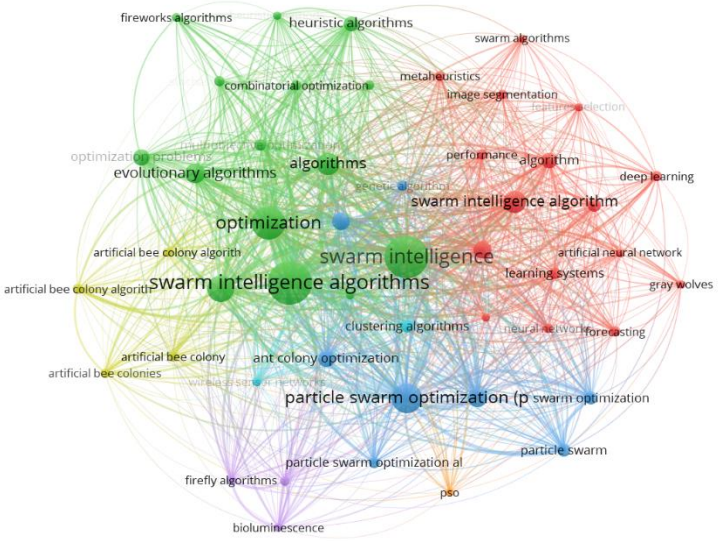

Figura 17: connessioni del Particle Swarm Optimization

Il funzionamento del PSO comincia con l'inizializzazione di una popolazione randomica di soluzioni candidate, chiamate particelle: ognuna di queste rappresenta una possibile soluzione nello spazio di ricerca del problema. La posizione di ciascuna particella è associata a una velocità, che indica la direzione e l'intensità dello spostamento della particella stessa nello spazio di ricerca (Tang et al., 2021).

L'algoritmo prosegue con l'aggiornamento delle posizioni e delle velocità delle particelle, secondo un modello matematico. Ogni particella ha una memoria della migliore posizione trovata fino a quel momento, denominata personal best (P_{best}), e viene influenzata anche dalla migliore posizione trovata da qualsiasi particella all'interno dello sciame, denominata global best (G_{best}).

La nuova velocità di una particella viene calcolata combinando tre componenti: la velocità precedente, l'attrazione verso P_{best} e l'attrazione verso G_{best}. Insieme alla velocità, di conseguenza si aggiorna anche la posizione. Questo processo continua iterativamente, con le particelle che si spostano nello spazio di ricerca seguendo una traiettoria influenzata sia dalle loro esperienze personali che dalle esperienze condivise dallo sciame (Qawqzeh et al., 2021).

Tuttavia, le particelle possono rimanere bloccate nella ricerca locale, riducendo la convergenza nel processo di ricerca. Per superare queste limitazioni, sono stati condotti diversi studi che hanno portato alla proposta di varianti modificate o ibride di questo algoritmo, riuscendo a migliorare le prestazioni.

L'Artificial Bee Colony Optimization (ABC) fu proposto per la prima volta da Karaboga nel 2015, e si basa sul comportamento delle api in una colonia. Le colonie sono composte da tre gruppi di api: le operaie, le osservatrici e le api esploratrici. Ogni tipo di ape ha il suo ruolo: quelle operaie, ad esempio, cercano casualmente una fonte di cibo, e tornano all'alveare con informazioni sulla stessa. Queste informazioni vengono condivise con le api osservatrici, che le valutano con un approccio probabilistico, per decidere se interessarsi alla fonte di cibo. La comunicazione avviene attraverso la "danza delle api", in cui l'ape danzante produce un suono ronzante muovendo il corpo da un lato all'altro per fornire informazioni (Tang et al., 2021).

L'algoritmo ABC ha una struttura di aggiornamento divisa in due fasi, che permette una rapida convergenza ai risultati migliori: questa è formata da esplorazione e sfruttamento.

L'esplorazione analizza l'intera area dello spazio di ricerca per trovare delle soluzioni promettenti, mentre lo sfruttamento valuta una porzione immediata del dominio per migliorare la soluzione.

Vengono richiesti tre parametri di controllo: numero di fonti di cibo, numero massimo di cicli e valore limite. All'inizio viene ipotizzata la fonte di cibo, che rappresenta la soluzione iniziale al problema di ottimizzazione. Successivamente, si determina la qualità del cibo, ovvero la qualità della soluzione associata. Le api osservatrici selezionano la fonte in base alla probabilità, ricavata dalla danza delle api.

Le api operaie, successivamente, modificano la soluzione per trovare soluzioni vicine e ne valutano la qualità, seguendo un processo in cui si confrontano la fonte di cibo attuale e quella vicina. Le api osservatrici ripetono quindi i passaggi di selezione, modifica e valutazione delle soluzioni vicine selezionate. Alla fine, le api esploratrici sostituiscono la fonte di cibo ormai abbandonata con una nuova soluzione, a seconda dei risultati (Kumoye et al., 2020).

In aggiunta, tra i più famosi e studiati algoritmi di SI, spicca sicuramente l'Ant Colony Optimization (ACO). Viene progettato per trovare una soluzione a problemi di ottimizzazione della pianificazione, simulando il comportamento di ricerca delle formiche.

Quando un gruppo di formiche deve trovare il percorso più breve tra la loro casa e una fonte di cibo, riesce a farlo con facilità anche senza l'uso di sensi come la vista. Questo è possibile grazie ai feromoni, sostanze chimiche secrete dalle formiche per comunicare. Inizialmente, le formiche scelgono un percorso casuale e, una volta raggiunto l'obiettivo, depositano feromoni lungo quel percorso.

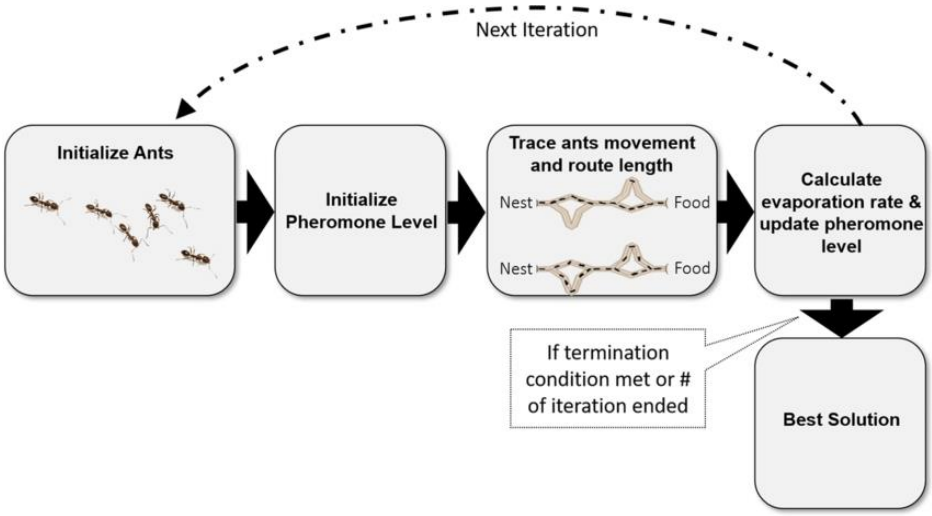

Figura 18: spiegazione grafica del funzionamento dell' ACO (Zehra et al., 2021)

Man mano che più formiche seguono questo percorso, la quantità di feromoni aumenta, rendendo più evidente il percorso migliore per le altre formiche attraverso un processo chiamato aggiornamento dei feromoni. Il percorso con la maggiore concentrazione di feromoni rappresenta il percorso più breve, mentre quello con la minore concentrazione rappresenta il percorso più lungo (Singh et al., 2021).

L'implementazione tecnica dell'ACO può essere descritta attraverso vari passaggi. In primo luogo, una colonia di formiche viene inizializzata con posizioni casuali. Ogni formica rappresenta una soluzione potenziale al problema di ottimizzazione.

Durante la ricerca, ognuno degli insetti costruisce una soluzione, muovendosi attraverso gli stati del problema secondo una probabilità influenzata dai feromoni e da una funzione euristica, che rappresenta la qualità della mossa. Dopo che tutte le formiche hanno completato la creazione delle loro soluzioni, il livello di feromoni sui percorsi viene aggiornato. Questo aggiornamento può includere sia l'evaporazione dei feromoni esistenti, che il deposito di nuovi feromoni da parte delle formiche che hanno trovato soluzioni buone.

Il processo iterativo continua fino a quando non viene soddisfatto un criterio di arresto, come il raggiungimento di un numero massimo di iterazioni o una soluzione sufficientemente buona. Tuttavia, l'algoritmo ACO può presentare problemi di stagnazione, dove tutte le formiche convergono verso la stessa soluzione, riducendo la diversità delle soluzioni esplorate e potenzialmente portando a soluzioni subottimali. Per affrontare queste limitazioni, sono stati sviluppati miglioramenti e varianti dell'ACO. Ad esempio, tecniche come il resetting dei feromoni, l'introduzione di formiche esploratrici che ignorano i feromoni esistenti, e l'uso di strategie di esplorazione e sfruttamento bilanciate sono state proposte per migliorare l'efficacia dell'algoritmo (Tang et al., 2021).

Altre tecnologie di Swarm Intelligence includono il Grey Wolf Optimizer (GWO), che verrà analizzato in seguito, il Bat Algorithm (BA), basato sulle frequenze scambiate dai pipistrelli, particolarmente efficace per problemi non lineari e multi-obiettivo; il Chicken Swarm Optimization (CSO), derivante dalle gerarchie e dalla socialità dei polli; il Lion optimization Algorithm (LOA), incentrato sulla difesa e sulla conquista territoriale dei leoni; e, infine, il Social Spider Algorithm (SSA), che sfrutta i diversi ruoli di maschio e femmina nella società dei ragni (Kaur & Kumar, 2020). Come si nota da quanto detto in precedenza, tutti gli algoritmi genetici nascono, e si strutturano, con un'idea comune, quella di trovare la soluzione partendo da una distribuzione casuale, seguendo una certa logica o modello matematico. Ciò che li rende così innovativi è la capacità di riuscire a ottimizzare qualsiasi tipo di analisi, conciliando al meglio la natura fisico-matematica del problema con il modello di ottimizzazione designato. La struttura dei diversi algoritmi, infatti, può essere più adatta per alcuni tipi di applicazione, o per altri; tuttavia, il grande numero di SI esistenti consente una vasta possibilità di scelta, permettendo, caso per caso, di trovare la migliore.

4.2 Grey Wolf Optimizer

L'algoritmo, scelto per la tesi in questione, basa la sua natura nel comportamento del lupo grigio durante la caccia, e nella sua gerarchia sociale. Il GWO, infatti, è un altro algoritmo di ispirazione biologica e sociale che si presta bene per simulare modelli fisico-matematici. Fino ad oggi, come sottolinea anche il nome dell'algoritmo, è stato utilizzato, come molti altre tecnologie SI, sotto forma di ottimizzatore, di processo o, alternativamente, degli iper-parametri di una rete neurale, ad esempio. Tuttavia, esso può esser adottato anche come tecnica principale in un modello regressivo, come nel caso di questa trattazione.

L'utilizzo del Grey Wolf Optimizer, in questo tipo di contesti, potrebbe non garantire i livelli di precisione delle reti neurali profonde o di altri metodi di DL, data la sua semplicità costruttiva e applicativa, ma consente all'utente di impostare la formulazione matematica utilizzata per predire il target, offrendo una conoscenza più completa e consapevole del problema. Ciò è molto utile, sia perché permette di modificare e personalizzare ogni tipo di applicazione (nel caso delle regressioni, una volta conosciute le leggi che regolano il variare del target), sia perché permette di modificare l'algoritmo in maniera specifica (Makhadmeh et al., 2024).

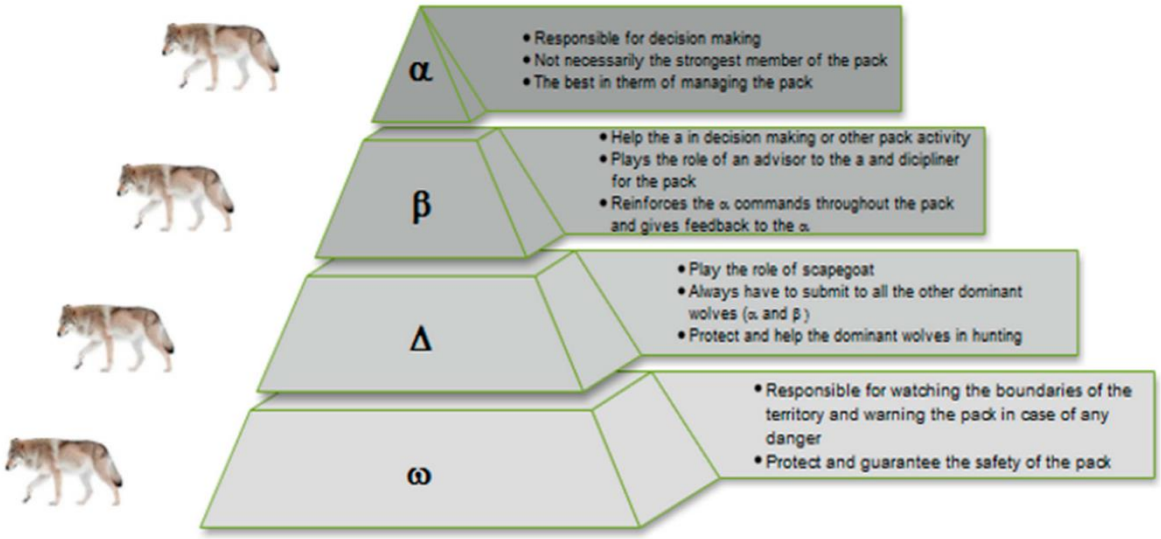

Figura 19: gerarchia dei lupi grigi nel branco (Kraiem et al., 2021)

Come si può notare dalla figura 16, il GWO si basa sulla struttura sociale e gerarchica osservata nei branchi di lupi grigi, membri della famiglia dei canidi, e predatori al vertice della catena alimentare. Vivendo tipicamente in branchi da cinque a dodici individui, i lupi grigi dimostrano un'elevata consapevolezza sociale, specialmente in termini di caccia e leadership.

Il lupo che guida il branco, sia maschio che femmina, è noto come Alpha. Esso prende decisioni cruciali riguardanti la caccia, i luoghi di riposo e i momenti di attività del branco. Anche se non è sempre il membro fisicamente più potente, l'Alpha eccelle nel guidare il gruppo, evidenziando come l'organizzazione e la disciplina siano più importanti della forza fisica (Liu et al., 2023).

Subito dopo troviamo i Beta, che fungono da subordinati diretti. Obbediscono all'Alpha, supportandolo nelle decisioni e nei compiti di gestione del branco, comandando i lupi di rango inferiore. Essi rafforzano le direttive dell'Alpha, agendo come suoi “consiglieri” e garanti della disciplina del branco.

I lupi che si sottomettono agli Alpha e ai Beta ma dominano i lupi di rango inferiore sono chiamati Delta. In questa categoria rientrano scout, sentinelle, anziani, cacciatori e assistenti, ognuno con un ruolo specifico all'interno del branco.

Il rango più basso, invece, è occupato dagli Omega, considerati i capri espiatori del branco. Gli omega devono cedere il passo a tutti i lupi dominanti e sono gli ultimi a mangiare. Spesso, gli omega svolgono anche il ruolo di babysitter per i cuccioli del branco.

Un aspetto importante del comportamento sociale dei lupi grigi è, come accennato in precedenza, la caccia di gruppo. Le principali fasi includono la localizzazione della preda, l'avvicinamento e il pedinamento, seguite dall'inseguimento e dal circondamento, che culminano con l'attacco.

Queste strategie di caccia collettiva sono la base per la modellazione dell'algoritmo di ottimizzazione in questione. La gerarchia di Alpha, Beta, Delta e Omega, infatti, viene utilizzata per simulare la ricerca e l'individuazione di soluzioni ottimali (Liu et al., 2023). Gli Alpha rappresentano le soluzioni migliori, seguite dai Beta, dai Delta e infine dagli Omega. Questa simulazione delle dinamiche di un branco di lupi consente al GWO di esplorare e sfruttare efficacemente lo spazio delle soluzioni, garantendo una convergenza positiva e applicazioni di successo in numerosi campi. Durante il ciclo collaborativo, viene valutata la distanza tra il gruppo Omega e i tre migliori lupi (Alpha, Beta, Delta). Successivamente, il GWO aggiorna le posizioni di questi lupi, cercando nuove soluzioni ottime (Dehghani et al., 2019a).

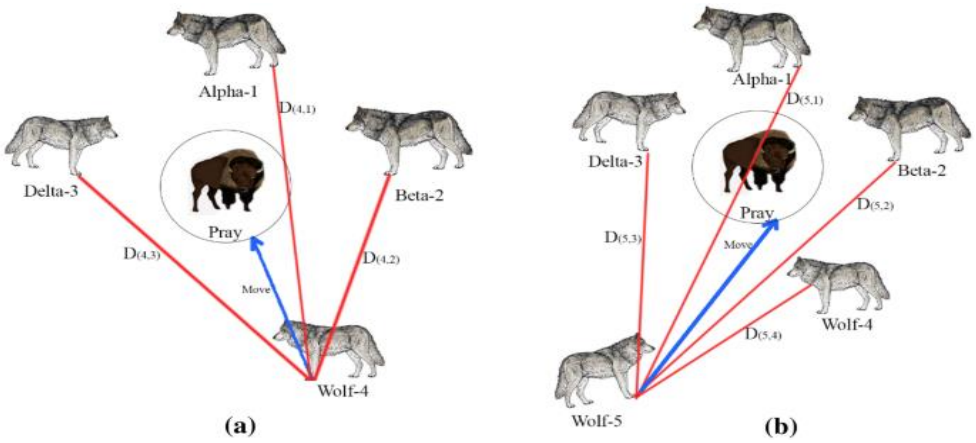

Figura 20: caccia del lupo grigio, applicata al GWO (Seyyedabbasi & Kiani, 2021)

L'implementazione matematica dell'algoritmo si basa, quindi, nel continuo aggiornamento delle posizioni dei lupi: i tre migliori lupi influenzano le future posizioni di tutti gli altri, come si vedrà in seguito. Per favorire il fattore stocastico del modello, invece, vengono aggiunti dei vettori e dei coefficienti, anch'essi analizzati in seguito.

Il GWO inizia il suo corso con le equazioni che rappresentano il circondamento della preda:

$$\vec{D} = |\vec{C} \cdot \vec{X}p(t) - X(t)|$$

$$\vec{X}(t+1) = \vec{X}p(t) - \vec{A} \cdot \vec{D}$$

dove $\vec{A}$ e $\vec{C}$ sono dei vettori dei coefficienti stocastici, $\vec{X}_p$ determina la posizione della preda, mentre $\vec{X}$ quella attuale del lupo. $\vec{D}$ è un vettore che specifica la nuova posizione del lupo, e t è il tempo di iterazione. I termini $\vec{C}$ e $\vec{A}$ sono formulati in questo modo:

$$\vec{A} = 2\vec{a} \cdot r_1 \rightarrow -\vec{a}$$

$$\vec{C} = 2 \cdot r_2$$

con $\vec{a}$ un coefficiente che rappresenta l'insieme dei vettori sull'iterazione che cambiano di valore linearmente da 2 a 0. D'altro canto, $\vec{r}_1$ e $\vec{r}_2$ rappresentano vettori casuali nell'intervallo [0, 1] (Dehghani et al., 2019b).

La caccia dei lupi grigi viene guidata dagli Alpha, mentre i Beta e i Delta contribuiscono a questo compito occasionalmente. Per la rappresentazione matematica della caccia, si assume che i migliori tre lupi abbiano una migliore conoscenza delle posizioni della preda. Pertanto, le soluzioni ottimali per le tre posizioni vengono ottenute come segue:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$

$$\vec{X}_1 = \vec{X}_\alpha - A_1 \cdot \vec{D}_\alpha$$

$$\vec{X}_2 = \vec{X}_\beta - A_2 \cdot \vec{D}_\beta$$

$$\vec{X}_3 = \vec{X}_\delta - A_3 \cdot \vec{D}_\delta$$

Il resto dei lupi, come mostrato sotto, seguirà gli altri e aggiornerà le proprie posizioni conseguentemente, tramite una media che tiene conto dei contributi dei tre migliori lupi (Dehghani et al., 2019b).

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

I lupi grigi terminano la caccia attaccando la preda quando questa smette di muoversi. Per rappresentare l'avvicinamento alla preda, viene ridotto il valore di $\vec{a}$. Si noti che l'intervallo di fluttuazione di $\vec{A}$ è legato ad $\vec{a}$. In altre parole, $\vec{A}$ è un valore casuale nell'intervallo $[-2a, 2a]$, dove a diminuisce da 2 a 0 nel corso delle iterazioni. Quando i valori casuali di $\vec{A}$ sono in $[-1, 1]$, la posizione successiva di un agente di ricerca può trovarsi in qualsiasi posizione tra la sua posizione attuale e la posizione della preda. Come detto, l'algoritmo GWO consente ai suoi agenti di ricerca di aggiornare la loro posizione in base alla posizione dell'alpha, beta e delta; e di attaccare la preda. Tuttavia, esso è incline a stagnare in soluzioni locali con questi operatori, e ha quindi bisogno di più operatori per enfatizzare l'esplorazione (Mirjalili et al., 2014).

Diventa necessario indurre una simulazione della divergenza dei membri del branco: si pone, quindi, il vettore $\vec{A}$ con valori casuali superiori all'unità, o inferiori all'unità negativa, per obbligare l'agente di ricerca a divergere dalla preda. Questo incrementa l'esplorazione e consente all'algoritmo di cercare in un dominio più ampio. Un altro componente del GWO che favorisce l'esplorazione è $\vec{C}$: esso, infatti, contiene valori casuali in [0, 2] e fornisce pesi randomici per la preda, in modo da enfatizzare stocasticamente ($\vec{C}$>1), o de-enfatizzare ($\vec{C}$<1), l'effetto della preda nel definire la distanza dalla posizione del lupo. Questo aiuta il GWO a mostrare un comportamento più casuale durante l'ottimizzazione, favorendo l'esplorazione e l'evitamento degli ottimi locali (Mirjalili et al., 2014). Il vettore $\vec{C}$ può essere considerato anche come l'effetto degli ostacoli nell'avvicinarsi alla preda in natura: in generale, gli ostacoli in natura appaiono nei percorsi di caccia dei lupi e, di fatto, impediscono loro di avvicinarsi rapidamente e comodamente alla preda. A seconda della posizione di un lupo, può dare casualmente alla preda un peso e renderla più difficile e lontana da raggiungere per i lupi, o viceversa.

In buona sostanza, a livello di implementazione, il GWO opera grazie a un vettore di coordinate, uno per ciascun lupo, che interagisce con il dataset in base al modello matematico assegnato, e che viene poi ottimizzato, come visto sopra, in base alle metriche scelte. Tutte queste equazioni, infatti, sono alla base del codice Python su cui è basato il modello regressivo di questa trattazione. Di seguito verrà analizzato lo script, descrivendo nel dettaglio ogni passaggio concettuale che viene effettuato.

4.3 Dataset e Preprocessing

Prima di analizzare nel dettaglio i passaggi fondamentali dell'algoritmo utilizzato per il progetto in questione, è necessario presentare il dataset preso in considerazione, e chiarire quali sono state le principali operazioni di Preprocessing adottate per far convergere i risultati.

Come accennato nel capitolo precedente, il dataset utilizzato per questa tesi era già stato lavorato da Song et al. nel 2020, attraverso una SVM e una ANN. Esso contiene un set di 10116 valori di solubilità di CO_2, misurati in soluzione con 124 diversi IL, ed è privo di valori mancanti.

Si trovano, all'interno del dataset, 51 colonne, che rappresentano la struttura molecolare di ogni liquido ionico: esse sono divise per cationi, anioni e sostituenti cationici; oltre a queste, troviamo anche due colonne con dentro dei valori, rispettivamente, di Temperatura e Pressione; parametri che influenzano direttamente la solubilità. Il dataset appare come segue:

Tabella 1: header del dataset

IL	Catione	Anione	X_CO2	T (K)	P (bar)	Comp. Strutturali es. [MIm]
[BMIM][BF4]	[BMIM]	[BF4]	valore	valore	valore	valore (intero)

La colonna target di Solubilità (S), chiamata ‘x_CO2’, contiene valori che vanno da 0.0000648 a 0.9516, in frazione molare. In questa colonna, analizzandola graficamente attraverso le funzioni della libreria Matplotlib, è stata evidenziata una maggior concentrazione di valori piccoli, vicini allo zero, a discapito di quelli vicini all’unità, con una distribuzione di valori piuttosto lineare.

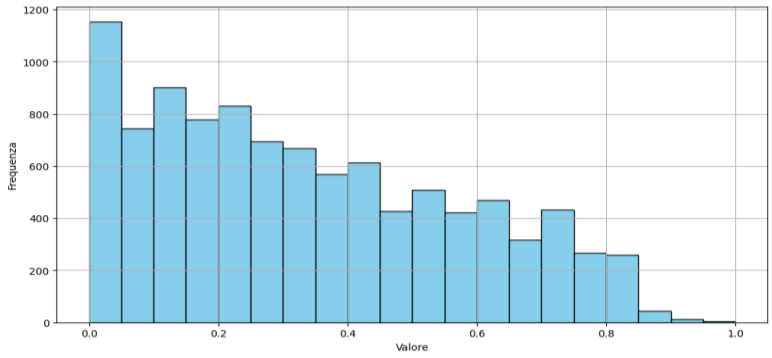

Figura 21: distribuzione dei valori nella colonna di Solubilità

Per quanto riguarda la Temperatura, invece, si nota che varia da 243.2 a 453.15 K, con valori sparsi in maniera piuttosto uniforme. D’altra parte, la Pressione, compresa tra 0.00798 e 499.9 bar, ha una spiccata concentrazione di valori bassi, mostrando una distribuzione ad esponenziale inversa. Ciò è molto importante da sottolineare, poiché, se dei valori piccoli, e molto vicini tra loro, apportano la stessa variazione di solubilità di due valori grandi e più lontani, questi vanno sicuramente pre-processati, ed è anche possibile intuire il tipo di modellazione matematica da dare al problema.

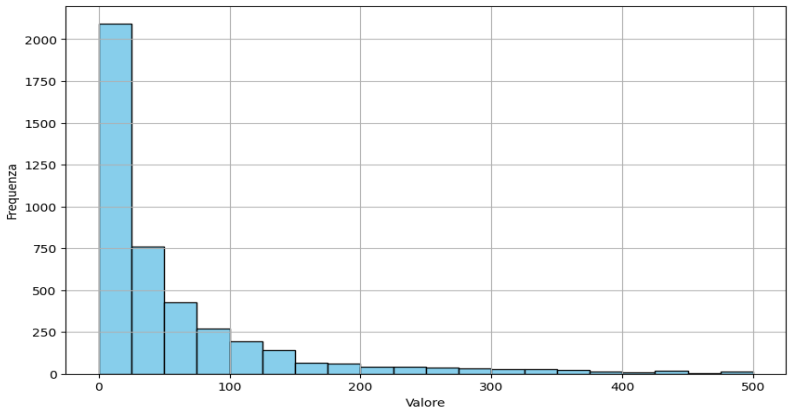

Figura 22: distribuzione dei valori nella colonna di Pressione

Infine, i componenti strutturali dei composti ionici, presentano valori interi, che variano da zero (quando un elemento non è presente nell'IL della specifica riga) fino a un massimo di 28, come nel caso della concentrazione di metilene (CH_2) all'interno del bis-trifluorometilsolfonil azanide, triesil-tetradecil fosfanio ([$P_{6,6,6,14}$] [Tf_2N]). Si evidenzia che la maggior parte dei componenti strutturali sono non nulli solo per le poche righe corrispondenti al IL di appartenenza, mentre solo qualche componente è presente in grandi quantità. Ad esempio, per i cationi e i sostituenti, i più presenti sono CH_3, CH_2 e MIm; mentre, relativamente alla parte degli anioni, troviamo Tf_2N, BF_4 e PF_6, con un numero superiore a 1000. Ciò suggerisce che questi elementi potranno esser presi in considerazione come features per la stima della solubilità. Per rafforzare questa teoria, è stata effettuata anche una analisi di correlazione, notando quali fossero le colonne più influenti su quella di Solubilità, riportando alle stesse conclusioni.

Come detto, uno degli aspetti positivi dell'utilizzo degli algoritmi di Swarm Intelligence, è, sicuramente, la conoscenza del modello matematico di base, che consente una serie di vantaggi sulla gestione del problema. Per poter impostare e ottimizzare il modello matematico, però, è necessario conoscere al meglio le relazioni che intercorrono tra le features e il target di un dataset.

In particolare, in questo caso è stato molto utile esplicitare le relazioni matematiche di Pressione e Temperatura, in funzione del target di Solubilità. I componenti strutturali, invece, vengono utilizzati solo come discriminante, visto che il numero che associa la loro quantità in uno specifico ione, non ha un significato fisico diretto. Proprio per questo, tra i coefficienti e la S, si è ipotizzata una relazione lineare.

Per estrarre le relazioni matematiche di cui sopra, si è proceduto graficando l'andamento di P rispetto a quello di S, in intervalli di dati a T circa costante, ripetendo la procedura per diversi IL. Anche in questo caso è stata utilizzata la libreria Matplotlib, ed è stato possibile ottenere dei grafici che delineano perfettamente l'andamento della Pressione.

Come si può notare dalle figure sottostanti, l'andamento delle curve assomiglia a una funzione logaritmica traslata verso l'alto, oppure a una quadratica, o a una cubica, a seconda dell'IL considerato e del range di T assegnato. Questo da un'indicazione molto importante sui possibili modelli matematici da assegnare all'algoritmo.

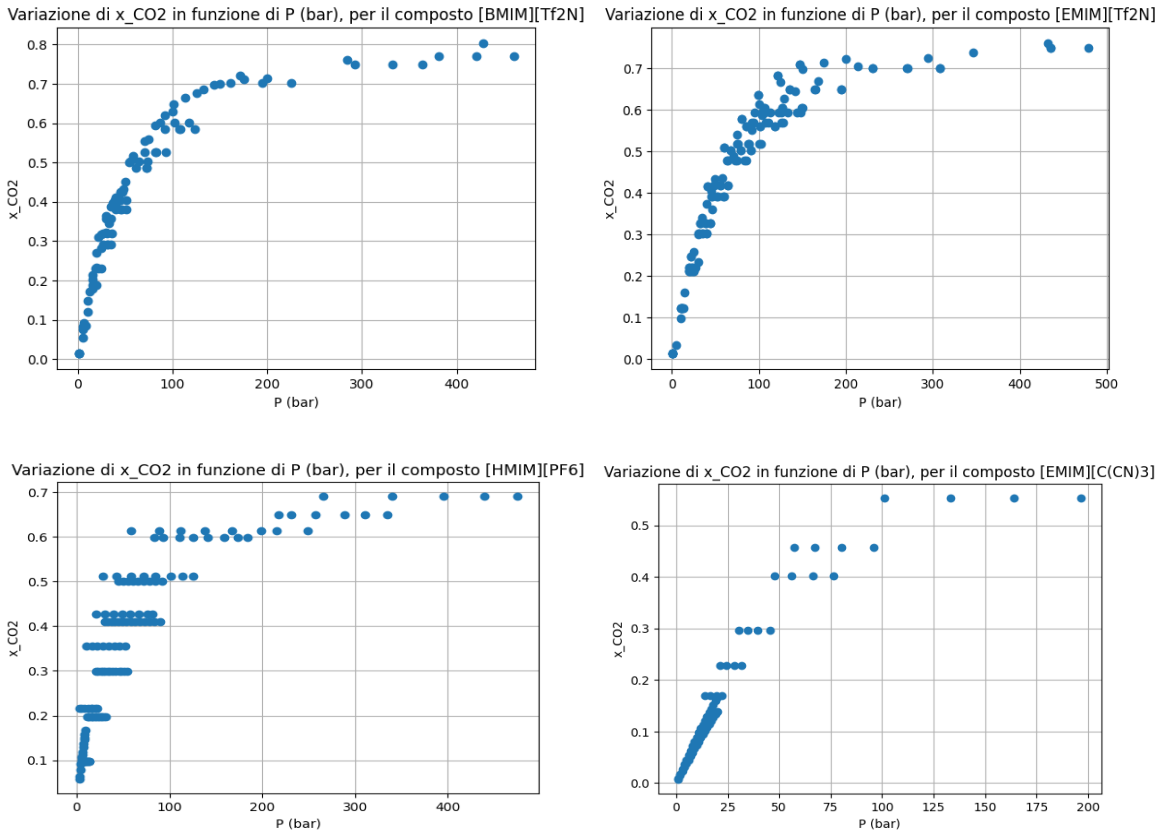

Figura 23: relazione tra Pressione e Solubilità

Lo stesso procedimento è stato applicato per la Temperatura, anche se scegliere dei range di P piuttosto costanti non è stato così semplice come per T, e ciò ha peggiorato la qualità dei grafici. Tuttavia, dalle immagini rimane evidente come tra T e S sussista una relazione di proporzionalità inversa. Si dovrà prevedere, quindi, che il termine riferito alla T, all'interno del modello matematico, venga posto al denominatore, cosicché per valori più alti si otterranno risultati più piccoli, e viceversa.

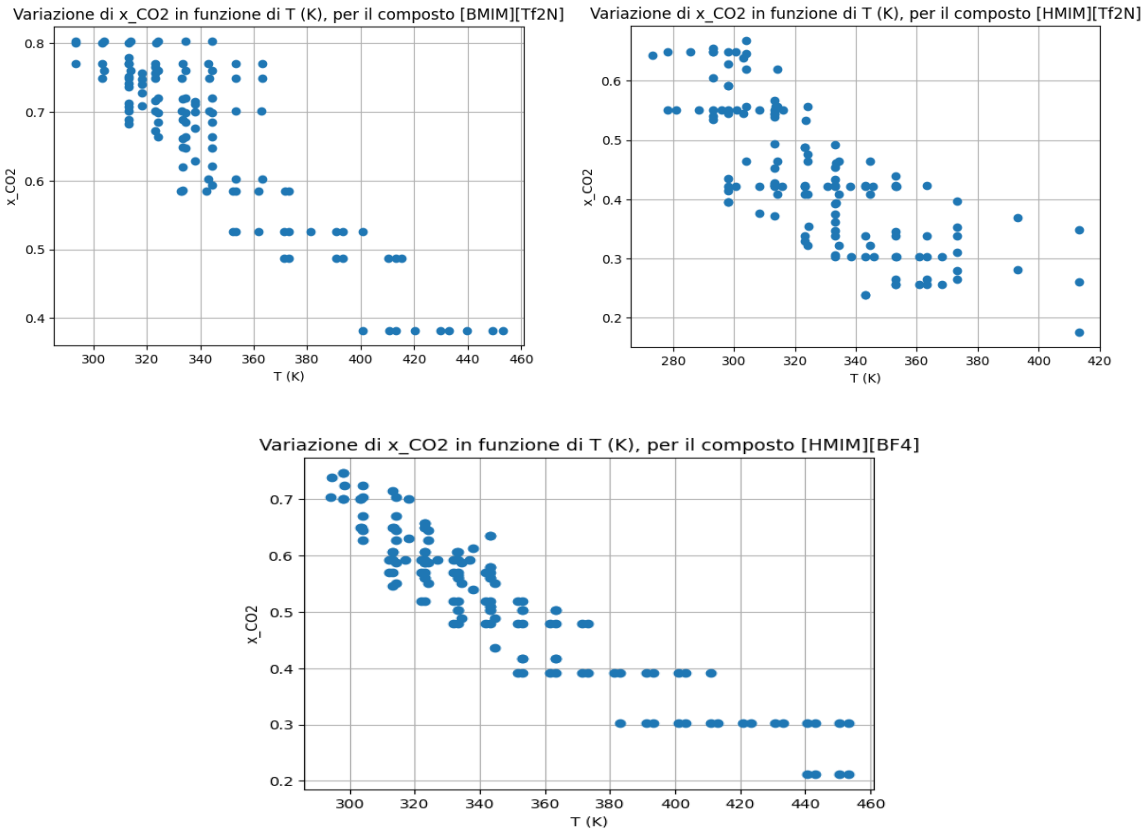

Figura 24: relazione tra Temperatura e Solubilità

Relativamente al Preprocessing, esso è un componente fondamentale nella preparazione dei dati all'interno di un algoritmo di AI, e descrive qualsiasi tipo di elaborazione effettuata su dati grezzi, per renderli adatti alle lavorazioni successive. La gestione dei dati è cruciale, poiché permette di fornire risultati precisi e robusti. I dati reali, infatti, possono essere spesso disordinati, poiché creati, elaborati e memorizzati da diverse persone, processi aziendali e applicazioni. Ciò risulta in dataset con campi mancanti, errori di input manuali, dati duplicati o nomi diversi per descrivere la stessa cosa.

Gli strumenti e i metodi di Preprocessing più comuni includono il campionamento, che seleziona un sottoinsieme rappresentativo da una grande popolazione di dati; la trasformazione, che manipola i dati grezzi per produrre un input standardizzato; il de-noising, che rimuove il rumore dai dati; e la normalizzazione, che organizza i dati per un accesso più efficiente.

In generale, gli algoritmi di machine learning e deep learning funzionano meglio quando i dati sono presentati in un formato che mette in evidenza gli aspetti rilevanti necessari per risolvere un problema. Le pratiche di feature engineering, che coinvolgono la riorganizzazione, la trasformazione e la riduzione dei dati, la selezione e la scalatura delle caratteristiche, aiutano a ristrutturare dataset grezzi in una forma adatta a particolari tipi di algoritmi, riducendo significativamente la potenza di elaborazione, e il tempo, necessari per addestrare un nuovo algoritmo o eseguire un'inferenza su di esso (Data Preprocessing: Definition, Key Steps and Concepts).

Gli elementi iniziali di Preprocessing di questo algoritmo hanno riguardato la trasformazione della colonna P. Come espresso in precedenza, infatti, essa presenta una massiva concentrazione di valori bassi: ciò implica che, per l'algoritmo, risulterebbe più difficile far apprezzare le differenze che una minima variazione di P apporta alla Solubilità, specie se tra valori molto piccoli.

Una delle prime operazioni effettuate, quindi, è stata una *feature engineering* per la P: si è tentato di, per così dire, "logaritmizzarla". Sostanzialmente, come si nota nella figura 22, è stata applicata la funzione *log1p*, di Numpy, ad ogni valore della colonna. Così facendo, i valori piccoli risultavano leggermente cresciuti e, soprattutto, distanziati, mentre quelli più grandi, diminuiti e avvicinati, rendendo la distribuzione di valori più omogenea. Anche altre operazioni, non di Preprocessing, come si vedrà in seguito, sono state tentate per poter sfruttare al meglio la feature di Pressione.

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np

#Seleziona la feature da scalare
feature_scaled = 'P (bar)'

#Calcola il logaritmo naturale dei valori della feature
df['log_' + feature_scaled] = np.log1p(df[feature_scaled])
```

Figura 25: codice relativo alla logaritmizzazione della Pressione

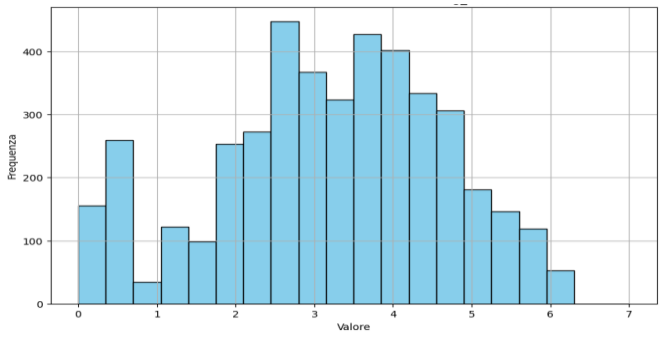

Figura 26: distribuzione dei valori nella colonna di Pressione logaritmizzata

Successivamente a ciò, è stato scelto di normalizzare i dati, attraverso la funzione *MinMaxScaler*, di SciKit-Learn; essa, infatti, riesce a mantenere le proporzioni dei dati originali, ma cambiando la scala e l'ordine di grandezza con cui questi vengono espressi.

Lo *scaler* è particolarmente utile quando i dati presentano range diversi, in quanto uniforma le caratteristiche su una stessa scala, migliorando le prestazioni degli algoritmi. *MinMaxScaler* funziona trasformando i dati secondo una formula che ridimensiona ogni valore, basandosi sul valore minimo e massimo di ciascuna caratteristica. Questa normalizzazione aiuta a ridurre l'impatto delle feature con varianze elevate e, di conseguenza, a migliorare la convergenza.

Attraverso il parametro *feature_range*, la funzione permette di personalizzare il range di valori all'interno del quale riportare i propri dati; di default questo intervallo va da 0 a 1, assegnando ai valori estremi trovati, proprio il valore nullo e l'unità. In questo caso, si è scelto di modificare il *feature_range* di default, escludendo gli estremi: si è cercato di evitare che il valore più piccolo di Pressione o di Temperatura fosse uguale a zero, poiché, così facendo, si riesce a mantenere le relazioni esistenti tra le features e il target. Inoltre, si è tentato di trovare una scala adatta per uniformare tutte le colonne.

Per quanto riguarda la Temperatura, avendo esplicitato la sua relazione di proporzionalità inversa con il target, non sarà sufficiente scalarla tra zero e uno. Infatti, seppur escludendo gli estremi, si avrebbero problemi con numeri troppo alti, tendenti all'infinito, dati i valori molto piccoli al denominatore.

Si è pensato, quindi, di scalare la T in un intervallo che andasse dal reciproco del valore più grande, fino al reciproco del valore più piccolo, del range di valori usato per le altre features. Infatti, scegliendo come estremo inferiore del dominio, un numero non eccessivamente piccolo, si garantisce che ci sia la stessa scala per tutte le caratteristiche, sebbene la Temperatura si trovi al denominatore.

```
#Si inizializza il MinMaxScaler con un intervallo personalizzato per la feature T
from sklearn.preprocessing import MinMaxScaler
scaler_A = MinMaxScaler(feature_range=(0.0001, 0.999))
scaler_B = MinMaxScaler(feature_range=(1/0.999, 1/0.0001))

#Si crea il primo dataset di features scalate
df_scaled_A = pd.DataFrame(scaler_A.fit_transform(df[['P (bar)','[CH3]','[CH2]','[Tf2N]']]),
                           columns=['P (bar)','[CH3]','[CH2]','[Tf2N]'])

#Si crea il dataset con la T scalata
df_scaled_B = pd.DataFrame(scaler_B.fit_transform(df[['T (K)']]), columns=['T (K)'])

#Combina le colonne scalate in un nuovo dataset
df_scaled = pd.concat([df_scaled_A, df_scaled_B], axis=1)

#Si inserisce la colonna T alla seconda posizione, per come è costruita la fitness function
colonna_da_spostare = df_scaled.pop('T (K)')
df_scaled.insert(1, 'T (K)', colonna_da_spostare)
```

Figura 27: codice relativo alla normalizzazione dei dati

Come si può notare dalla figura 24, ciò è stato implementato dividendo la colonna T dal resto del dataset, scalandola separatamente, per poi riunirla, attraverso la funzione *concat*, della libreria Pandas. Sono stati scelti, poi, diversi *feature_range*, per provare quale fosse il migliore per l'intero dataset analizzato con delle specifiche features, o per il cluster preso in analisi.

4.4 Implementazione dell'algoritmo

In questa sezione vengono analizzate nel dettaglio tutte le operazioni effettuate per far convergere i dati verso i risultati migliori, e, soprattutto, per poter riadattare un ottimizzatore ad algoritmo di regressione principale. Come delineato in precedenza, infatti, il GWO nasce come ottimizzatore, ma, con alcune accortezze, può ben funzionare anche in quest'altra versione. La funzione alla base dell'algoritmo, in questo caso, sarà strutturata in funzione del legame tra features e target, e verrà utilizzata per predire valori di Solubilità, da confrontare con quelli reali. La posizione di ogni lupo verrà valutata in base ai rispettivi valori predetti, calcolando la "distanza" dalla preda, ovvero l'errore tra il dato reale e quello stimato.

Nel funzionamento classico dell'algoritmo, le coordinate dei lupi rappresentano le variabili della funzione da ottimizzare, che qui sono i valori delle diverse colonne. In questo tipo di applicazione, invece, gli elementi di ogni vettore posizione rappresentano i coefficienti moltiplicativi della funzione.

Per quanto detto in precedenza, sarà necessario, ovviamente, un indicatore che permetta di valutare le posizioni dei lupi. La scelta della metrica è ricaduta sul Root Mean Square Error (RMSE), data la possibilità di apprezzare al meglio anche i più piccoli miglioramenti ottenuti. Questo indice di accuratezza è particolarmente utile, oltre che nella regressione, in ambiti come la previsione del tempo e il riconoscimento di pattern, e in generale in qualsiasi campo in cui è importante valutare la precisione delle previsioni di un modello.

Il calcolo del RMSE avviene attraverso vari passaggi. Inizialmente, per ciascuna osservazione (riga) nel dataset, si calcola l'errore assoluto, ossia la differenza tra il valore osservato e il valore previsto dal modello. Questo viene elevato al quadrato, per eliminare i segni negativi ed evidenziare maggiormente gli errori più grandi. Successivamente, si calcola la media di tutti questi quadrati, per poi generarne la radice, e riportare l'unità di misura degli errori alla scala originale.

Un altro aspetto fondamentale per la regressione è quello che ci permette di capire se, il modello addestrato, sarà valido anche per atri dati: la divisione del dataset in train set e test set. Questa tecnica consente di verificare se l'algoritmo è in grado di generalizzare al meglio su dati nuovi e non visti, evitando anche il problema dell'overfitting, che capita quando il modello si adatta troppo bene ai dati di addestramento, ma fallisce con dati esterni. Una suddivisione comune è quella dell'80-20, dove l'80% dei dati viene utilizzato per l'addestramento e il 20% per il test.

La divisione dei dati può essere eseguita grazie alla libreria SciKit-Learn, che offre la funzione *train_test_split* per dividere facilmente il dataset in set di addestramento e test, permettendo di specificare la proporzione dei dati da destinare a ciascun set.

Per la sua implementazione pratica, il codice dell'algoritmo Grey Wolf Optimizer viene coadiuvato da due elementi fondamentali: la classe lupo, che in questo caso è stata chiamata MyWolf, e la Fitness Function (FF), modello matematico su cui si basa il problema. Di seguito verranno presentate e analizzate nel dettaglio ognuna di queste istanze, insieme al corpo centrale dell'algoritmo.

4.4.1 Fitness Function

La Fitness Function in un algoritmo GWO rappresenta, come anticipato, il modello matematico che regola la classificazione dei lupi, tramite una funzione con tante variabili quante sono le coordinate di ciascun elemento. Tutti i modelli matematici implementati si servono delle librerie Numpy, per la manipolazione di array e operazioni numeriche; SciKit-Learn, per calcolare l'errore quadratico medio; e Math, per calcolare le radici necessarie. Per questa trattazione sono state pensate diverse FF, ma solo 2 hanno portato a risultati accettabili, e verranno analizzate in seguito.

La prima funzione implementata è stata chiamata *T_function*, essendo una funzione lineare contenente un solo elemento fratto per la colonna di Temperatura (legata da proporzionalità inversa con il target). Per questa funzione, infatti, è stata utilizzata la feature di Pressione logaritmizzata, descritta precedentemente. Nella figura sottostante, viene mostrato il codice relativo alla FF in questione.

```python
#Funzione Lineare con elemento fratto per la Temperatura
import numpy as np
from sklearn.metrics import mean_squared_error
from math import sqrt

def T_function(position, dataset1, dataset2):
    num_rows = len(dataset1)
    num_dimensions = len(position)
    predicted_values = np.zeros(num_rows)

    for i in range(num_rows):
        # Calcola la combinazione lineare
        linear_combination = 0
        for j in range(num_dimensions):
            if j == 1:  # Applica la proporzione inversa al secondo elemento
                linear_combination += position[j] / dataset1.iloc[i, j]
            else:
                linear_combination += dataset1.iloc[i, j] * position[j]

        predicted_values[i] = linear_combination

    actual_values = dataset2
    rmse = np.sqrt(np.mean((actual_values - predicted_values) ** 2))

    return rmse
```

Figura 28: *T_function*

Come si nota dalla riga di definizione della funzione, essa prende tre argomenti: position, ovvero le coordinate del lupo corrente; il dataset, con le features di input; e la colonna dei valori di Solubilità corrispondenti alle righe del primo dataset.

Successivamente inizia lo script della funzione, che inizializza tre variabili: *num_rows*, che memorizza il numero di righe in *dataset1*; *num_dimensions*, che rappresenta il numero di dimensioni (coefficienti) del vettore posizione dei lupi; e *predicted_values*, inizializzato come un array di zeri della stessa lunghezza del numero di righe nel primo dataset, che conterrà i valori di S, predetti dalla funzione.

Si fa partire un ciclo che itera su ogni riga di *dataset1*. La variabile *linear_combination* viene posta a zero, per calcolare la combinazione lineare degli input. All'interno del ciclo, ce n'è un altro annidato che itera su ogni dimensione della *position*.

Quando si incontrano i valori di Temperatura, posti nella seconda colonna, viene applicata una proporzionalità inversa, dividendo il coefficiente corrispondente per il valore dell'elemento nella riga i di *dataset1*. Per tutte le altre dimensioni, viene calcolata la combinazione lineare moltiplicando il valore della feature per il coefficiente corrispondente, aggiungendolo a *linear_combination*.

Dopo aver calcolato la combinazione lineare per ogni riga, il valore risultante viene memorizzato in *predicted_values*, fino ad arrivare a completarle tutte. Di contro, *actual_values* memorizza i valori reali dei dati target dal *dataset2*, sotto forma di lista.

Negli ultimi passaggi, viene calcolata la differenza tra i valori reali e quelli predetti, poi elevata al quadrato. Si ottiene, quindi, la media di questi valori e se ne estrae la radice quadrata. Infine, la funzione restituisce il valore del RMSE come misura della fitness per l'algoritmo GWO. A livello matematico, quindi, la funzione opera come segue per ogni riga:

$$S = \sum_{i=1, i\neq 2}^{n} l_i f_i + \frac{l_2}{f_2}$$

Dove S è la solubilità, *l* il vettore delle coordinate del lupo e *f* rappresenta le features, che cambiano riga per riga.

L'altra FF implementata nell'algoritmo è stata chiamata *P_cubic_function*, e segue esattamente la stessa logica della *T_function*, ma aggiunge un elemento radice cubica per la Pressione. In questo caso, infatti, la colonna di Pressione viene trattata come le altre features, e viene data in pasto all'algoritmo esattamente com'è, tenendo conto della sua relazione con la Solubilità.

```python
#Funzione con elemento radice cubica per la Pressione
import numpy as np
from sklearn.metrics import mean_squared_error
from math import sqrt

def P_cubic_function(position, dataset1, dataset2):
    num_rows = len(dataset1)
    num_dimensions = len(position)
    predicted_values = np.zeros(num_rows)

    for i in range(num_rows):
        # Calcola la combinazione lineare
        linear_combination = 0
        for j in range(num_dimensions):
            if j == 0:  # Applica la radice cubica solo al primo elemento
                linear_combination += np.cbrt(dataset1.iloc[i, j]) * position[j]
            elif j == 1:  # Applica la proporzione inversa al secondo elemento
                linear_combination += position[j] / dataset1.iloc[i, j]
            else:
                linear_combination += dataset1.iloc[i, j] * position[j]

        predicted_values[i] = linear_combination

    actual_values = dataset2
    rmse = np.sqrt(np.mean((actual_values - predicted_values) ** 2))
    return rmse
```

Figura 29: Funzione *P_cubic_function*

Come si nota dalla prima riga del ciclo, la funzione calcola la radice cubica dei valori nella prima colonna (dov'è posta la P) del *dataset1*, tramite *cbrt* di Numpy. Questo valore viene poi moltiplicato per il coefficiente corrispondente in *position*, e aggiunto a *linear_combination*. Ciò significa che, per la colonna di P, invece di utilizzare il valore direttamente, viene utilizzata la radice cubica del valore, linearizzando la relazione che intercorre tra la Pressione e la Solubilità.

Avendo a disposizione solo dei grafici che predicono la relazione tra P ed S, questa non può essere esplicitata analiticamente, e, di conseguenza, sono state implementate altre funzioni, simili alla *P_cubic_function*, con elementi radice quadrata e quarta, invece di quella cubica. Queste alternative, però, non hanno portato a risultati altrettanto buoni, e, pertanto, sono state scartate. In questo caso la funzione matematica rimane la stessa, ma si aggiunge il termine di radice cubica al primo termine.

$$S = \sqrt[3]{f_1} \cdot l_1 + \frac{f_2}{l_2} + \sum_{i=3}^{n} f_i \cdot l_i$$

4.4.2 Classe MyWolf

Si passa, ora, a descrivere la classe lupo, chiamata *MyWolf*, che è stata utilizzata come agente all'interno dell'algoritmo. La classe è progettata per inizializzare un lupo con una posizione casuale nello spazio di ricerca, calcolare il suo valore di fitness, e gestire i dataset necessari per la valutazione degli agenti.

Dopo aver importato i moduli necessari, si parte col definire il costruttore della classe, *__init__*. Esso è addetto a creare nuovi lupi, ricevendo cinque parametri: innanzitutto la FF, poi il numero di dimensioni, coerente con il numero delle features scelte; il seme, per generare numeri casuali, e i dataset del problema.

```
import random
import numpy as np
from sklearn.metrics import mean_squared_error
from math import sqrt

class MyWolf:
    def __init__(self, fitness_function, dim, seed, dataset1, dataset2):
        self.rnd = random.Random(seed)
        self.position = [0.0 for _ in range(dim)]


        #Inizializza la posizione del lupo con valori casuali
        for i in range(dim):
            self.position[i] = self.rnd.random()

        # Memorizza i dataset necessari per la fitness function
        self.dataset1 = dataset1
        self.dataset2 = dataset2

        # Calcola il valore di fitness utilizzando la nuova fitness function
        self.fitness = fitness_function(self.position, dataset1, dataset2)
```

Figura 30: Classe *MyWolf*

Successivamente, *self.rnd* inizializza un generatore di numeri casuali con il seme fornito, mentre *self.position* crea una lista di zeri, con una lunghezza specificata dal numero delle dimensioni, rappresentando la posizione iniziale del lupo. Il seme viene specificato all'interno del corpo centrale dell'algoritmo, in modo tale da garantire la riproducibilità dell'esperimento.

Viene, poi, impostato ogni elemento della posizione del lupo a un valore casuale tra 0 e 1. Nelle righe successive, invece, si memorizzano *dataset1* e *dataset2* come attributi della classe. La stessa cosa avviene, successivamente, per il valore di fitness del lupo, utilizzando la FF fornita e la posizione corrente del lupo, insieme ai dataset. Il risultato viene memorizzato nell'attributo *self.fitness*.

4.4.3 Algoritmo

Il corpo dell'algoritmo GWO viene spiegato in seguito, mostrando i passaggi chiave, già introdotti precedentemente, attraverso l'implementazione Python della logica di funzionamento. A livello tecnico, l'algoritmo è rappresentato da una funzione, chiamata *regressor_gwo*, che prende sei parametri di input: la funzione di fitness, il numero massimo di iterazioni, il numero di elementi nella popolazione, le features scelte, e i due dataset utilizzati nella FF. Viene anche creato un oggetto Random con un seme fisso, che, come detto, consente per la riproducibilità dei risultati.

```
import random
import copy

def gwo(fitness, max_iter, n, dim, dataset1, dataset2):
    rnd = random.Random(0)
    #si creano n lupi casuali
    population = [
        MyWolf(fitness, dim, seed=i, dataset1=dataset1, dataset2=dataset2)
        for i in range(n)
    ]

    #si ordina la popolazione sulla base del fitness value
    population = sorted(population, key=lambda temp: temp.fitness)

    #le 3 migliori soluzioni sono chiamate alpha, beta e gamma
    alpha_wolf, beta_wolf, gamma_wolf = copy.copy(population[:3])
```

Figura 31: foto del GWO fino alla classificazione dei lupi

Come riporta il commento del codice, si parte creano la popolazione, con ciascun elemento in una posizione casuale. La classe *MyWolf* è utilizzata per generare questi lupi, e ognuno viene inizializzato con un seme univoco e i dataset forniti.

Conseguentemente, tutti gli *n* lupi sono ordinati in base ai valori di fitness, in ordine crescente. Questo permette di identificare i migliori tre, nominati *alpha_wolf*, *beta_wolf* e *delta_wolf*. Di seguito, viene mostrato il ciclo principale del GWO, che continuerà fino a raggiungere il numero massimo di iterazioni.

```
#ciclo principale del GWO
Iter = 0
while Iter < max_iter:
    if Iter % 1 == 0 and Iter >= 1:
        print("Iter="+str(Iter)+"best fitness = %.3f" % alpha_wolf.fitness)

    #a descresce linearmente da 2 a 0
    a = 2 * (1 - (Iter / max_iter))

    #aggiorna la posizione dei lupi con quelle dei 3 migliori
    for i in range(n):
      A1, A2, A3 = a * (2 * rnd.random() - 1), a * (
              2 * rnd.random() - 1), a * (2 * rnd.random() - 1)
      C1, C2, C3 = 2 * rnd.random(), 2 * rnd.random(), 2 * rnd.random()
      X1 = [0.0 for i in range(dim)]
      X2 = [0.0 for i in range(dim)]
      X3 = [0.0 for i in range(dim)]
      Xnew = [0.0 for _ in range(dim)]
      for j in range(dim):
        X1[j] = alpha_wolf.position[j] - A1 * abs(
          C1 * alpha_wolf.position[j] - population[i].position[j])
        X2[j] = beta_wolf.position[j] - A2 * abs(
          C2 *  beta_wolf.position[j] - population[i].position[j])
        X3[j] = gamma_wolf.position[j] - A3 * abs(
          C3 * gamma_wolf.position[j] - population[i].position[j])
        Xnew[j] += X1[j] + X2[j] + X3[j]

      for j in range(dim):
            Xnew[j]/=3.0
```

Figura 32: ciclo principale del GWO

Si introduce il parametro *a*, che controlla l'intensità della ricerca esplorativa, decrementando il suo valore linearmente da 2 a 0 nel corso delle iterazioni. Grazie ad *a*, vengono calcolati i vettori *A1*, *A2*, *A3* e *C1*, *C2*, *C3*. Questi ultimi sono utilizzati per aggiornare le posizioni dei lupi, dove *A* controlla la distanza di esplorazione e *C* introduce variabilità nella direzione della ricerca.

Si ottengono, poi, per ogni lupo, i contributi della nuova posizione (*X1*, *X2*, *X3*), influenzati rispettivamente dalle posizioni di Alpha, Beta e Delta. Fatto ciò, la posizione *Xnew* equivarrà alla media dei contributi.

Se la nuova posizione ha un valore di fitness migliore rispetto a quella corrente, viene effettuata una selezione *greedy*: la posizione del lupo, e il suo valore di fitness associato, vengono aggiornati. Infine, la popolazione viene riordinata, e verranno rinominati Alpha, Beta e Delta.

Terminate tutte le iterazioni, la funzione restituisce la posizione del miglior lupo, che rappresenta la soluzione ottimale trovata dall'algoritmo GWO. Per mandare in esecuzione il codice, bisognerà scrivere una riga che ci permette di assegnare tutti gli input necessari all'algoritmo. Il risultato ottenuto sarà, quindi, un vettore di coefficienti, corrispondenti alle coordinate del lupo, ciascuna per ogni feature.

```
        #calcolo del valore di fitness
        fnew = fitness(Xnew, dataset1, dataset2)

        #selezione greedy
        if fnew < population[i].fitness:
            population[i].position = Xnew
            population[i].fitness = fnew

    #classificazione basata sulle posizioni aggiornate
    population = sorted(population, key=lambda temp: temp.fitness)

    alpha_wolf, beta_wolf, gamma_wolf = copy.deepcopy(population[:3])

    Iter += 1

#restituzione della soluzione migliore
return alpha_wolf.position
```

Figura 33: termine del ciclo GWO

Una volta ottenuto il vettore di coefficienti, e il suo valore di fitness, lo si applica direttamente alle colonne di feature del test, con lo stesso modello matematico della FF. Ciò ci permette di ottenere un vettore di valori predetti, da cui poi scaturisce il valore dell'RMSE, grazie alla comparazione con i valori di S del set di test. Infine, si decide di graficare l'andamento dei valori predetti, sovrapponendolo a quello dei valori reali di Solubilità. Ciò permetterà di effettuare una critica analisi, visiva e analitica, apprezzando le similitudini e le differenze tra i dati stimati e quelli effettivi. Tutto questo, insieme alle varie prove eseguite, verrà esposto e analizzato nel capitolo successivo.

5 Risultati

Ricapitolando, con questo progetto si è voluto realizzare un algoritmo di SI adatto alla predizione di un target. Nell'applicazione specifica, esso riesce a stimare i valori di Solubilità della CO_2 all'interno di soluzioni con Liquidi Ionici, basandosi sulle leggi fisiche legano il target con la Pressione, la Temperatura e i componenti strutturali dei IL. Gli elementi principali che sono stati realizzati sono: le Fitness Functions, in grado di rappresentare il modello matematico adottato; la classe *MyWolf*, che definisce e inizializza gli agenti di ricerca; e, infine, la personalizzazione del corpo centrale del GWO.

Per ottenere un numero significativo di risultati comparabili, l'algoritmo è stato eseguito più volte e con diverse combinazioni di parametri, al fine di identificare i più efficaci. Tenendo conto delle considerazioni precedenti, sono state implementate tutte le strade più approfondite, sia nella gestione delle features, che nel Preprocessing dei dati.

In particolare, per la feature di Temperatura, abbiamo applicato una normalizzazione inversa e reciproca rispetto a quella utilizzata per le altre caratteristiche. Questo garantisce una scala uniforme, evitando problemi legati al denominatore. Per quanto riguarda la Pressione, si utilizza la funzione *T_function* se la colonna è stata logaritmizzata. In caso contrario, viene impiegata la funzione *P_cubic_function*. Le caratteristiche, infine, sono state scalate in un intervallo compreso tra 0.0001 e 0.999, che, dopo diversi tentativi, si è rivelato essere il più efficace.

Durante le analisi si è notato, inoltre, che la parte anionica dei composti risulta essere presente con coefficienti pari a zero e uno. Si è pensato, come anticipato, di clusterizzare il dataset, ogni parte per un determinato composto anionico. Ad esempio, si nota che l'intero dataset contiene 4301 righe con presenza dell'anione Tf_2N: sono state effettuate, quindi, delle prove che mostrano le previsioni solamente per le righe con questo composto. Lo stesso procedimento è stato applicato per gli anioni BF_4, PF_6, $C(CN)_3$ e TfO, presenti per più di cinquecento righe ciascuno.

Per le features aggiuntive a T ed S, ovvero i cationi ed i sostituenti, si è visto che considerarle in numero minore possibile permette di migliorare i risultati. Di conseguenza, sono stati utilizzati solamente CH_2 e CH_3, che sono i sostituenti cationici più presenti, e permettono di fare da discriminante per tutti quanti i composti.

In aggiunta, per quanto riguarda gli iper-parametri, dopo svariati tentativi sono stati presi come riferimento un numero di iterazioni pari a 30, e una popolazione di lupi con 90 membri, trovando un compromesso tra le prestazioni e il carico computazionale. Di seguito vengono riportati gli andamenti dei dati stimati rispetto a quelli previsti, caso per caso.

Partendo dalle prove effettuate con il dataset nella sua interezza, si è pensato di aggiungere la feature Tf_2N, per fare da discriminante ulteriore, soprattutto per quanto riguardasse la parte anionica; esso è, infatti, l'anione più presente nel dataset. Nella versione con la radice cubica della Pressione, si ottiene un RMSE per il test di set pari a 0.126, mentre nel set di train vale 0.125. Con la feature logaritmica, invece, i dati sono pari a 0.129 per il train set e di 0.131 per il test set.

Per il cluster relativo all'anione Tf_2N, nella regressione con la funzione cubica, la metrica riporta un valore di 0.112 per il set di test, e di 0.111 per il set di train. Nel caso opposto, troviamo un RMSE pari a 0.113 per il set di test, e 0.108 per il set di train.

Per mostrare al meglio i risultati, in dei grafici leggibili, nei primi quattro grafici (due per l'intero dataset e due per il cluster Tf_2N) verrà effettuata un'operazione di smoothing, applicando una finestra di quattro valori per la media mobile. Ciò permette di apprezzare al meglio gli andamenti, nonostante i grafici non siano da prendere come indicatore oggettivo, poiché contiene dei valori mediati. In tutte le figure il grafico blu rappresenterà i valori reali, mentre quello arancione, quelli predetti.

Figura 34: Dataset intero, P logaritmizzata

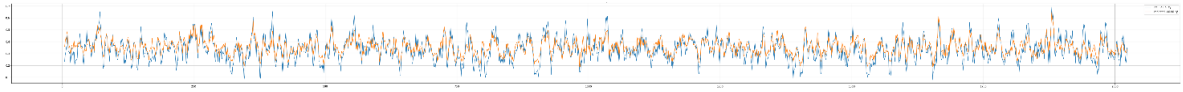

Figura 35: Dataset intero, *P_cubic_function*

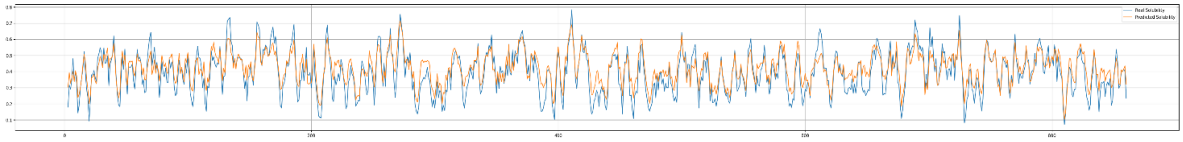

Figura 36: Cluster Tf2N, P logaritmizzata

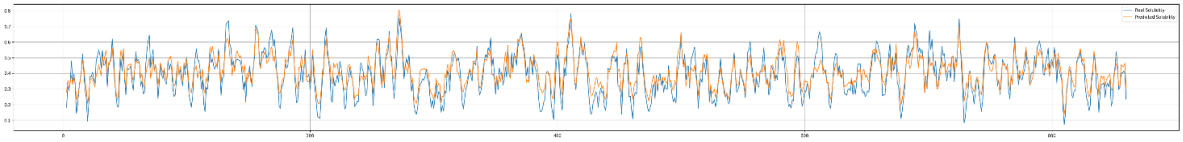

Figura 77: Cluster Tf2N, *P_cubic_function*

Nella sezione riguardante il gruppo triflato (TfO), si ottiene l'RMSE per il set di test che vale 0.075, mentre per il train 0.084, con la *T_function*. Utilizzando la *P_cubic_function*, invece, i valori sono di 0.084 per il test e 0.091 per il train. Si nota come, in questo caso, il test mostri addirittura un abbassamento della metrica rispetto al train set, aspetto molto positivo. Il cluster $C(CN)_3$ mostra un comportamento ancora migliore, restituendo 0.075 e 0.068 nel primo caso; mentre con la Pressione sotto radice cubica, 0.080 e 0.076.

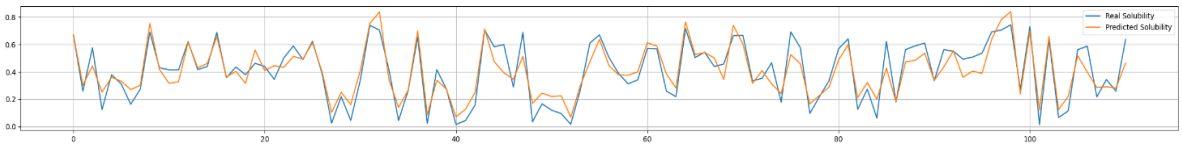

Figura 38: Cluster TfO, *P_cubic_function*

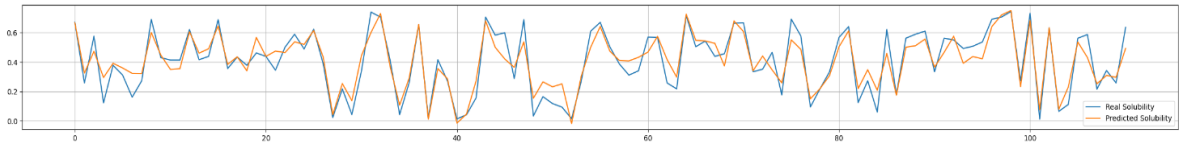

Figura 39: Cluster TfO, P logaritmizzata

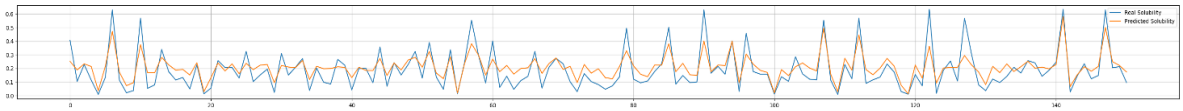

Figura 40: Cluster C(CN)3, *P_cubic_function*

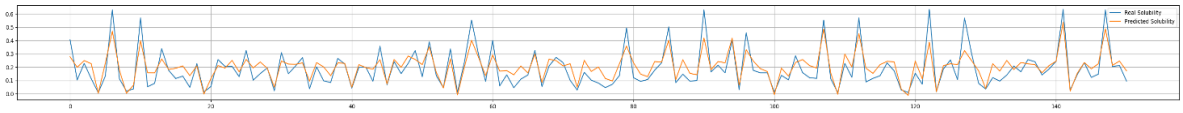

Figura 41: Cluster C(CN)3, P logaritmizzata

I cluster BF_4 e PF_6, lunghi entrambi 1116 righe, presentano dei valori con andamento simile. Il primo mostra un RMSE che vale 0.92 nel test e 0.095 nel train con *T_function*, viceversa un RMSE di 0.090 nel test e di 0.096 nel train con *P_cubic_function*. L'esafluorofosfato (PF_6), invece, riporta RMSE 0.086 nel test e 0.087 nel train per il primo caso; quando la Pressione viene presa al cubo, troviamo 0.090 per il set di addestramento e 0.088 per quello di test.

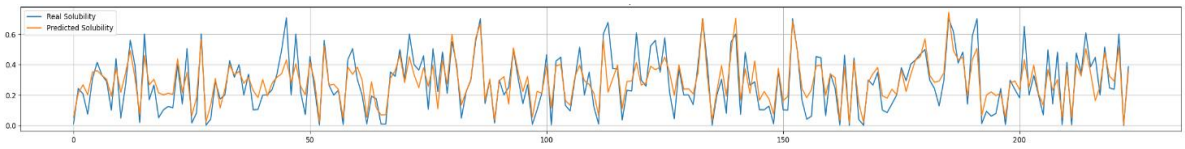

Figura 428: Cluster BF4, *P_cubic_function*

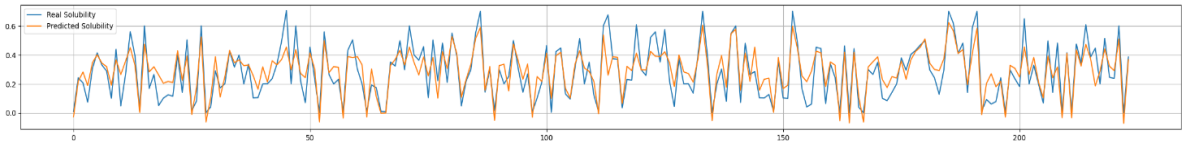

Figura 43: Cluster BF4, P logaritmizzata

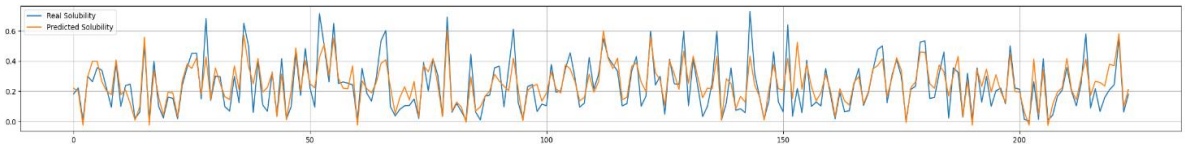

Figura 44: Cluster PF6, P logaritmizzata

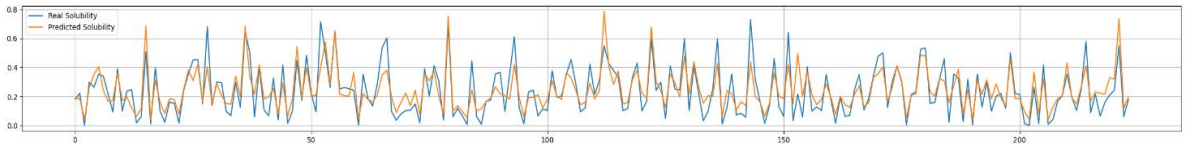

Figura 45: Cluster PF6, *P_cubic_function*

6 Conclusioni e sviluppi futuri

Come si può notare dai grafici e dai valori di RMSE riportati, le modalità di analisi, che differiscono solo per la gestione della feature di Pressione, hanno prestazioni simili, diventando leggermente migliori l'una dell'altra, a seconda del cluster analizzato. In generale, si può affermare che la logaritmizzazione della P consenta di raggiungere picchi, soprattutto per valori bassi, che l'elevamento a potenza non riesce a dare. Parallelamente, però, la maggior parte dei cluster, tramite la funzione *P_cubic_function*, riesce a stimare in maniera più accurata un maggior numero di valori di Solubilità. Questa soluzione rimane più adatta per set di dati abbastanza omogenei, dove è ben definita la relazione con il target.

Molto importante è stata la divisione del dataset in cluster di anioni. Ripetendo il procedimento per ogni porzione di dataset riferita ad uno specifico anione, infatti, questo passaggio permette di predire la Solubilità per ogni IL a disposizione, con tempistiche di esecuzione minori ed accuratezza maggiore. La clusterizzazione si è rivelato un elemento fondamentale, che ha permesso di raggiungere ottimi risultati.

Rimane chiaro, come è stato anticipato, e anche valutando i valori ottenuti in questo elaborato, che il GWO non raggiunge i livelli di precisione delle reti neurali profonde, o di altri metodi di DL, ma permette di acquisire un elevato grado di consapevolezza e comprensione del modello che si sta analizzando, consentendo di addentrarsi di più nella dinamica del problema.

Con gli algoritmi di SI, infatti, riuscendo ad approfondire il più possibile le leggi matematiche che regolano il problema, si possono apportare sempre più miglioramenti al modello, rendendolo di volta in volta più preciso. In questa analisi, ad esempio, se la relazione con la Temperatura venisse esplicitata in maniera univoca e chiara, cosa che è risultata difficile, si potrebbe apportare da subito un miglioramento nella FF, aumentando le capacità predittive dell'algoritmo.

Il modello sviluppato per questa trattazione potrebbe essere utilizzato in qualsiasi tipo di applicazione di carattere regressivo, specialmente laddove si conosca il comportamento fisico-matematico del problema. Per quanto concerne l'applicazione specifica, sicuramente, progredendo con la ricerca, il modello potrà essere ulteriormente migliorato, consentendo di inserirlo in vari processi di CCS, per ottimizzare l'assorbimento della CO_2.

Bibliografia e sitografia

7 Applications of Machine Learning in Manufacturing in 2023. (n.d.). Retrieved June 17, 2024, from https://pixelplex.io/blog/machine-learning-in-manufacturing/

10 Companies Using Machine Learning in Cool Ways. (n.d.). Retrieved July 4, 2024, from https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications

A Guide to Digital Twin Development - Visartech Blog. (n.d.). Retrieved July 4, 2024, from https://www.visartech.com/blog/digital-twin-solution-development-guide/

Absorption or Adsorption? | Durpro. (n.d.). Retrieved July 4, 2024, from https://www.durpro.com/en_US/blog/faq-5/absorption-or-adsorption-402

Akinola, T. E., Bonilla Prado, P. L., & Wang, M. (2022). Experimental studies, molecular simulation and process modelling\simulation of adsorption-based post-combustion carbon capture for power plants: A state-of-the-art review. *Applied Energy*, *317*, 119156. https://doi.org/10.1016/J.APENERGY.2022.119156

Alalaiwat, D., & Khan, E. (2024). Post-combustion carbon capture process modeling, simulation, and assessment of synergistic effect of solvents. *International Journal of Greenhouse Gas Control*, *135*, 104145. https://doi.org/10.1016/J.IJGGC.2024.104145

Baskaran, D., Saravanan, P., Nagarajan, L., & Byun, H. S. (2024). An overview of technologies for capturing, storing, and utilizing carbon dioxide: Technology readiness, large-scale demonstration, and cost. *Chemical Engineering Journal*, *491*, 151998. https://doi.org/10.1016/J.CEJ.2024.151998

Carter, A., Imtiaz, S., & Naterer, G. F. (2023). Review of interpretable machine learning for process industries. *Process Safety and Environmental Protection*, *170*, 647–659. https://doi.org/10.1016/J.PSEP.2022.12.018

Castro-Pardo, S., Bhattacharyya, S., Yadav, R. M., de Carvalho Teixeira, A. P., Campos Mata, M. A., Prasankumar, T., Kabbani, M. A., Kibria, M. G., Xu, T., Roy, S., & Ajayan, P. M. (2022). A comprehensive overview of carbon dioxide capture: From

materials, methods to industrial status. *Materials Today*, *60*, 227–270. https://doi.org/10.1016/J.MATTOD.2022.08.018

Data Preprocessing: Definition, Key Steps and Concepts. (n.d.). Retrieved June 26, 2024, from https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing

Davoodi, S., Al-Shargabi, M., Wood, D. A., Rukavishnikov, V. S., & Minaev, K. M. (2023). Review of technological progress in carbon dioxide capture, storage, and utilization. *Gas Science and Engineering*, *117*, 205070. https://doi.org/10.1016/J.JGSCE.2023.205070

Dehghani, M., Riahi-Madvar, H., Hooshyaripor, F., Mosavi, A., Shamshirband, S., Zavadskas, E. K., & Chau, K. wing. (2019a). Prediction of Hydropower Generation Using Grey Wolf Optimization Adaptive Neuro-Fuzzy Inference System. *Energies 2019, Vol. 12, Page 289*, *12*(2), 289. https://doi.org/10.3390/EN12020289

Dehghani, M., Riahi-Madvar, H., Hooshyaripor, F., Mosavi, A., Shamshirband, S., Zavadskas, E. K., & Chau, K. wing. (2019b). Prediction of Hydropower Generation Using Grey Wolf Optimization Adaptive Neuro-Fuzzy Inference System. *Energies 2019, Vol. 12, Page 289*, *12*(2), 289. https://doi.org/10.3390/EN12020289

Faisal Elmobarak, W., Almomani, F., Tawalbeh, M., Al-Othman, A., Martis, R., & Rasool, K. (2023). Current status of CO2 capture with ionic liquids: Development and progress. *Fuel*, *344*, 128102. https://doi.org/10.1016/J.FUEL.2023.128102

Freitas, D., Lopes, L. G., & Morgado-Dias, F. (2020). Particle Swarm Optimisation: A Historical Review Up to the Current Developments. *Entropy 2020, Vol. 22, Page 362*, *22*(3), 362. https://doi.org/10.3390/E22030362

Gaijon, P., Kant, A., Ghosh, S., & Singh, M. R. (2022). Progressive function of ionic liquids in polymer chemistry. *Advanced Applications of Ionic Liquids*, 479–495. https://doi.org/10.1016/B978-0-323-99921-2.00010-0

Hong, W. Y. (2022). A techno-economic review on carbon capture, utilisation and storage systems for achieving a net-zero CO2 emissions future. *Carbon Capture Science & Technology*, *3*, 100044. https://doi.org/10.1016/J.CCST.2022.100044

Hosseinpour, M., Shojaei, M. J., Salimi, M., & Amidpour, M. (2023). Machine learning in absorption-based post-combustion carbon capture systems: A state-of-the-art review. *Fuel*, *353*, 129265. https://doi.org/10.1016/J.FUEL.2023.129265

Kaur, K., & Kumar, Y. (2020). Swarm Intelligence and its applications towards Various Computing: A Systematic Review. *Proceedings of International Conference on Intelligent Engineering and Management, ICIEM 2020*, 57–62. https://doi.org/10.1109/ICIEM48762.2020.9160177

Kraiem, H., Aymen, F., Yahya, L., Triviño, A., Alharthi, M., & Ghoneim, S. S. M. (2021). A Comparison between Particle Swarm and Grey Wolf Optimization Algorithms for Improving the Battery Autonomy in a Photovoltaic System. *Applied Sciences 2021, Vol. 11, Page 7732*, *11*(16), 7732. https://doi.org/10.3390/APP11167732

Kumoye, A. O., Prasad, R., & Fonkam, M. (2020). Swarm Intelligence Algorithm and its Application:A Critical Review. *2020 International Conference in Mathematics, Computer Engineering and Computer Science, ICMCECS 2020*. https://doi.org/10.1109/ICMCECS47690.2020.246996

Kuroki, N., Suzuki, Y., Kodama, D., Alam Chowdhury, F., Yamada, H., & Mori, H. (2023). Machine Learning-Boosted Design of Ionic Liquids for CO 2 Absorption and Experimental Verification. *J. Phys. Chem. B*, *127*. https://doi.org/10.1021/acs.jpcb.2c07305

Liu, Y., As'arry, A., Hassan, M. K., Hairuddin, A. A., & Mohamad, H. (2023). Review of the grey wolf optimization algorithm: variants and applications. *Neural Computing and Applications 2023 36:6*, *36*(6), 2713–2735. https://doi.org/10.1007/S00521-023-09202-8

Ma, J., Li, L., Wang, H., Du, Y., Ma, J., Zhang, X., & Wang, Z. (2022). Carbon Capture and Storage: History and the Road Ahead. *Engineering*, *14*, 33–43. https://doi.org/10.1016/J.ENG.2021.11.024

Madejski, P., Chmiel, K., Subramanian, N., & Kuś, T. (2022). Methods and Techniques for CO2 Capture: Review of Potential Solutions and Applications in Modern Energy

Technologies. *Energies 2022, Vol. 15, Page 887*, *15*(3), 887. https://doi.org/10.3390/EN15030887

Makhadmeh, S. N., Al-Betar, M. A., Doush, I. A., Awadallah, M. A., Kassaymeh, S., Mirjalili, S., & Zitar, R. A. (2024). Recent Advances in Grey Wolf Optimizer, its Versions and Applications: Review. *IEEE Access*, *12*, 22991–23028. https://doi.org/10.1109/ACCESS.2023.3304889

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, *69*, 46–61. https://doi.org/10.1016/J.ADVENGSOFT.2013.12.007

Petrovic, B., Gorbounov, M., & Masoudi Soltani, S. (2022). Impact of Surface Functional Groups and Their Introduction Methods on the Mechanisms of CO2 Adsorption on Porous Carbonaceous Adsorbents. *Carbon Capture Science and Technology*, *3*, 100045. https://doi.org/10.1016/j.ccst.2022.100045

Podder, J., Patra, B. R., Pattnaik, F., Nanda, S., & Dalai, A. K. (2023). A Review of Carbon Capture and Valorization Technologies. *Energies 2023, Vol. 16, Page 2589*, *16*(6), 2589. https://doi.org/10.3390/EN16062589

Priya, A. K., Devarajan, B., Alagumalai, A., & Song, H. (2023). Artificial intelligence enabled carbon capture: A review. *Science of The Total Environment*, *886*, 163913. https://doi.org/10.1016/J.SCITOTENV.2023.163913

Qawqzeh, Y., Alharbi, M. T., Jaradat, A., & Sattar, K. N. A. (2021). A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments. *PeerJ Computer Science*, *7*, e696. https://doi.org/10.7717/PEERJ-CS.696

Ratnakar, R. R., Chaubey, V., & Dindoruk, B. (2023). A novel computational strategy to estimate CO2 solubility in brine solutions for CCUS applications. *Applied Energy*, *342*, 121134. https://doi.org/10.1016/J.APENERGY.2023.121134

Risso, R., Cardona, L., Archetti, M., Lossani, F., Bosio, B., & Bove, D. (2023). A Review of On-Board Carbon Capture and Storage Techniques: Solutions to the 2030 IMO

Regulations. *Energies 2023, Vol. 16, Page 6748*, *16*(18), 6748. https://doi.org/10.3390/EN16186748

Seyyedabbasi, A., & Kiani, F. (2021). I-GWO and Ex-GWO: improved algorithms of the Grey Wolf Optimizer to solve global optimization problems. *Engineering with Computers*, *37*(1), 509–532. https://doi.org/10.1007/S00366-019-00837-7/FIGURES/14

Singh, H., Tyagi, S., Kumar, P., Gill, S. S., & Buyya, R. (2021). Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions. *Simulation Modelling Practice and Theory*, *111*, 102353. https://doi.org/10.1016/J.SIMPAT.2021.102353

Song, Z., Shi, H., Zhang, X., & Zhou, T. (2020). Prediction of CO2 solubility in ionic liquids using machine learning methods. *Chemical Engineering Science*, *223*, 115752. https://doi.org/10.1016/J.CES.2020.115752

Sun, J., Sato, Y., Sakai, Y., & Kansha, Y. (2023a). A review of ionic liquids and deep eutectic solvents design for CO2 capture with machine learning. *Journal of Cleaner Production*, *414*, 137695. https://doi.org/10.1016/J.JCLEPRO.2023.137695

Sun, J., Sato, Y., Sakai, Y., & Kansha, Y. (2023b). A review of ionic liquids and deep eutectic solvents design for CO2 capture with machine learning. *Journal of Cleaner Production*, *414*, 137695. https://doi.org/10.1016/J.JCLEPRO.2023.137695

Tang, J., Liu, G., & Pan, Q. (2021). A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends. *IEEE/CAA Journal of Automatica Sinica*, *8*(10), 1627–1643. https://doi.org/10.1109/JAS.2021.1004129

Yang, Z., Chen, B., Chen, H., & Li, H. (2023a). A critical review on machine-learning-assisted screening and design of effective sorbents for carbon dioxide (CO2) capture. *Frontiers in Energy Research*, *10*. https://doi.org/10.3389/FENRG.2022.1043064

Yang, Z., Chen, B., Chen, H., & Li, H. (2023b). A critical review on machine-learning-assisted screening and design of effective sorbents for carbon dioxide (CO2) capture.

Frontiers in Energy Research, *10*, 1043064. https://doi.org/10.3389/FENRG.2022.1043064/BIBTEX

Yao, J., Han, H., Yang, Y., Song, Y., & Li, G. (2023). A Review of Recent Progress of Carbon Capture, Utilization, and Storage (CCUS) in China. *Applied Sciences 2023, Vol. 13, Page 1169*, *13*(2), 1169. https://doi.org/10.3390/APP13021169

Yu Tsivadze, A., Aksyutin, O. E., Ishkov, A. G., -, al, Yang, J., Ni, W., Ruan, B., Yang, C., Qi, J., Wang, A., Zha, J., Liu, C., & Yao, S. (2023). Application of machine learning in MOFs for gas adsorption and separation. *Materials Research Express*, *10*(12), 122001. https://doi.org/10.1088/2053-1591/AD0C07

Zehra, S. S., Qureshi, R., Dev, K., Shahid, S., & Bhatti, N. A. (2021). Comparative Analysis of Bio-Inspired Algorithms for Underwater Wireless Sensor Networks. *Wireless Personal Communications*, *116*(2), 1311–1323. https://doi.org/10.1007/S11277-020-07418-8