POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Gestionale

Tesi di Laurea Magistrale

Smart Waving: un applicazione del reinforcement learning nella generazione dei task di picking

Relatore:
Prof. Rafele Carlo

Laureando:
Beccherle Filippo

Anno Accademico: 2023/2024

Abstract

Questo documento esamina l'applicazione dell'intelligenza artificiale nel processo di generazione dei task di picking, proponendo una soluzione a una delle attività più time-consuming nella gestione del magazzino. Vengono analizzate nel dettaglio diverse possibili applicazioni dell'intelligenza artificiale all'interno della catena logistica di un'azienda, con particolare attenzione all'integrazione di un modello di reinforcement learning nel processo di prelievo.

L'algoritmo di reinforcement learning proposto è progettato per determinare autonomamente il momento ottimale per pianificare un ordine di picking, generando task di pianificazione tali da minimizzare la distanza percorsa dal picker durante le operazioni.

Il documento si conclude con una riflessione sui risultati ottenuti e una presentazione dei possibili sviluppi futuri del progetto.

Indice

Introduzione

Questa tesi esplora lo sviluppo di un modello di reinforcement learning per ottimizzare la pianificazione dei task di picking, concentrandosi su alcuni casi studio che dimostrino i vantaggi derivanti dall'integrazione di tale strumento nel processo di planning aziendale. Il lavoro è strutturato in modo da fornire un quadro completo della tecnologia, delle applicazioni e dei risultati ottenuti.

Inizialmente, i capitoli introduttivi presentano l'argomento, definendo la terminologia e le nozioni di base, introducendo il concetto di intelligenza artificiale, analizzando le varie branche della materia e studiandone le possibili applicazioni alla catena logistica di un'azienda; inoltre, è stato inquadrato il problema del picking, studiando le possibili soluzioni attualmente utilizzate e gli usi più diffusi dell'intelligenza artificiale all'interno di questo ambito.

Successivamente, si approfondisce il processo di sviluppo del modello "Smart waving". In particolare, il terzo capitolo introduce l'azienda che ha ospitato il progetto e descrive le esigenze specifiche di ottimizzazione del picking. Nei capitoli successivi, l'attenzione si sposta sugli aspetti di progettazione, sviluppo, e training del modello di reinforcement learning, fornendo una descrizione dettagliata delle scelte progettuali, delle funzioni chiave e del processo di fine-tuning.

Infine, la tesi discute i risultati ottenuti, confrontando le performance del modello con le tecniche tradizionali di pianificazione. Conclude con una riflessione sulle limitazioni del modello attuale e con una proposta di possibili sviluppi futuri per rendere lo Smart Waving uno strumento scalabile e adattabile a diversi contesti aziendali.

Il progetto portato avanti ha visto la partecipazione di competenze differenziate. In particolare, io ho potuto mettere a disposizione del team le mie capacità in ambito di ricerca e analisi di fattibilità, nel design degli elementi costitutivi del modello, nella fase di testing e fine-tuning dei parametri utilizzati e nella valutazione dei risultati ottenuti. Le mie analisi durante la fase di fine-tuning hanno permesso di ottimizzare i parametri tecnici, migliorando la stabilità del modello e garantendo risultati più consistenti.

Lo sviluppo del codice dell'algoritmo, tuttavia, ha richiesto il supporto da parte di figure tecniche del team con maggiore esperienza in questo campo, in particolare per la scrittura del codice principale e la configurazione degli ambienti di simulazione. Questa collaborazione ha permesso di ottenere un modello che rappresenta una solida base per ulteriori sviluppi. Il progetto, frutto di una stretta sinergia tra diverse competenze, apre interessanti prospettive per la logistica avanzata, delineate nei capitoli finali.

Capitolo 1

Intelligenza artificiale: definizione e principali applicazioni alla supply chain di un'azienda

L'intelligenza artificiale (IA) sta rivoluzionando il mondo industriale e non solo, portando con sè un cambiamento radicale sia nella modalità di esecuzione dei task più semplici, sia nel modo di concepire i processi aziendali. L'IA si pone come obiettivo lo sviluppo di "macchine intelligenti", ovvero sistemi che siano in grado di replicare il comportamento umano quando posti di fronte a problemi complessi, che richiedono un processo razionale di risoluzione da parte di colui che li affronta. Alcuni esempi includono il riconoscimento di schemi, il processo decisionale e l'apprendimento autonomo.
Il processo di innovazione tecnologica iniziato con l'avvento di questa nuova tecnologia ha rivoluzionato il ruolo dell'uomo all'interno dell'azienda: i compiti più ripetitivi e manuali vengono, sempre più spesso, delegati ai robot, mentre l'essere umano assume il ruolo di supervisore. Tale cambiamento sta avvenendo all'interno di ogni settore industriale, a partire dalle grandi aziende. Parallelamente, l'IA sta avendo un impatto notevole nel processo di decision making, la disponibilità di tools basati sull'intelligenza artificiale supporta il manager nel processo decisionale, consentendo di avere ampia visibilità su tutte le informazioni necessarie e garantendo la possibilità di simulare diversi scenari, valutando l'impatto di ciascuno di essi sulla profittabilità e l'efficienza dell'azienda.
L'IA è una tecnologia estremamente versatile, dunque, gli ambiti di applicazione sono infiniti, spaziando dall'efficientamento della produzione all'utilizzo di robot intelligenti nel processo di picking, passando per algoritmi per la previsione della domanda. I vantaggi derivanti dall'introduzione di sistemi basati sull'IA sono numerosi, non solo in termini di ritorni economici e riduzione di costi, ma anche in termini di efficienza operativa, miglioramento nella qualità dei prodotti, senza trascurare i benefici nelle condizioni di

lavoro del personale. Tuttavia, l'intelligenza artificiale porta con sè anche diverse sfide. La preoccupazione principale legata a questi sistemi è dovuta alla loro scarsa capacità di discernere tra cio che è etico e ciò che non lo è, tra ciò che è giusto e ciò che è sbagliato, per questa ragione i sistemi regolatori devono stilare delle normative chiare, con il fine di definire i confini etici e di responsabilità nell'uso di queste tecnologie [1].

Secondo uno studio pubblicato da Gartner, azienda leader nella consulenza strategica, entro il 2028 avverrano due radicali cambiamenti all'interno del mondo delle operations [2]:

1. Circa il 25% dei KPI verranno elaborati da modelli di intelligenza artificiale generativa.
2. Gli "smart robots" supereranno numericamente gli esseri umani nei settori del manufacturing, del retail e della logistica.

Queste statistiche evidenziano come l'IA assumerà un ruolo strategico di primo piano negli anni a venire e le aziende dovranno attrezzarsi adeguatamente, in modo da non farsi trovare impreparate. Sempre secondo Gartner, attualmente solo il 10% dei CEO utilizza l'intelligenza artificiale in modo strategico, e queste aziende tendono a definire KPI già nella fase di sviluppo dei vari use case dell'IA [2]. I dati riportati sottolineano l'importanza che ha la capacità di un'azienda ad adattarsi ai continui cambiamenti nel contesto in cui opera per poter mantenere la posizione all'interno di un mercato in costante evoluzione.

Tra i settori maggiormente trasformati dall'IA spicca il supply chain management, ovvero le varie metodologie utilizzate per la gestione della catena logistica di un'azienda, dalla selezione e valutazione di nuovi fornitori mediante un vendor rating system, all'ottimizzazione dei trasporti, fino alla pianificazione della produzione mediante MRP e MPS. La supply chain riveste un ruolo cruciale nella competitività aziendale, rappresentando un elemento chiave per garantire un detemrinato livello di servizio ai clienti. Conseguentemente, tutte le scelte hanno un impatto sul prodotto finale e su come questo verrà percepito dai consumatori. È quindi fondamentale considerare attentamente il trade-off tra livello di servizio che si vuole offrire al cliente e costi da sostenere: un fornitore più efficiente, ad esempio, garantirà la disponibilità dei prodotti in tempi brevi, consentendo consegne più rapide ai clienti; d'altra parte, un network diffuso permetterà di avere dei magazzini in prossimità dei consumatori, permettendo una contrazione del delivery time, ma generando costi operativi maggiori e sacrificando, in parte, l'efficienza. Questi sono solo due esempi delle numerose decisioni da prendere nella configurazione di un network logistico. Ogni scelta ha un impatto diverso sul livello di servizio e sui costi, rendendo fondamentale una gestione efficace della supply chain.

L'intelligenza artificiale è un concetto estremamente ampio, costituito da vari ambiti, ognuno con caratteristiche e applicazioni specifiche. Nel contesto di questo documento, verranno analizzate tre principali tecniche di IA e il loro possibile utilizzo all'interno della supply chain: **machine learning (ML)**, **deep learning (DL)** e **reinforcement learning (RL)**. Il machine learning si focalizza sull'addestramento di algoritmi adatti all'analisi dei dati per la ricerca di pattern e a fare previsioni o decisioni automatizzate; il deep learning è una sotto-branca del ML, utilizza reti neurali complesse per esaminare grandi volumi di dati non strutturati; il reinforcement learning (RL) è utile per la risoluzione di problemi dinamici e decisionali, dove un agente viene addestrato tramite interazioni con l'ambiente. Queste metodologie verranno analizzate nel dettaglio nel capitolo seguente.

1.1 Machine learning

Il **machine learning (ML)** nasce dal tentativo di replicare la capacità degli esseri umani di apprendere dall'esperienza. L'obiettivo di questa disciplina è lo sviluppo di algoritmi di apprendimento in grado di costruire dei modelli a partire da un set di dati fornitogli. Il processo di creazione di questi modelli è noto come *apprendimento* o *addestramento*, durante il quale viene utilizzato un dataset "per imparare" e creare regole che permettano di effettuare previsioni basate su nuovi dati. Lo scopo del ML è quello di individuare pattern o schemi all'interno dei dati, per garantire maggiore accurattezza nelle predizioni fatte [3].
A differenza dell'intelligenza artificiale tradizionale, che si basa su inferenze logiche (ad esempio, se a porta a b e b porta a c, allora a porta a c), il machine learning adotta un approccio statistico guidato dai dati. Partendo dalle osservazioni fornite in input e output, il modello è in grado di sviluppare una funzione che mappa le relazioni tra i dati stessi [4]. Questo approccio permette al ML di adattarsi a scenari complessi e dinamici, dove le relazioni tra variabili non sono sempre lineari o intuitive.

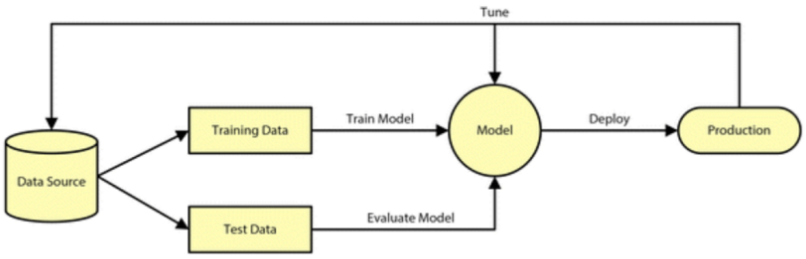

Figure 2: Supervised Machine learning workflow [23]

Figura 1.1: Schema concettuale del funzionamento di un algoritmo di machine learning.

1.1.1 Tipologie di machine learning

Esistono numerose categorie di machine learning, ciascuna caratterizzata da specifici ambiti di applicazione e metodologie. Per lo scopo di questo documento, verranno discusse le due principali categorie di machine learning: supervised learning e unsupervised learning, che rappresentano le tecniche più diffuse.

Tipologia di learning	**Algoritmo**	**Descrizione**
Supervised learning	Decision tree	Using related values, decision trees (DTs) will categorize attributes in different groups which can be applied for classification purposes.
	Support vector machine	Lavorando sul calcolo dei margini, la Support Vector Machine (SVM) può essere applicata al meglio per la classificazione.
	Supervised neural network	Utilizzando una SNN (Supervised Neural Network), l'uscita prevista e quella effettiva vengono confrontate e i parametri vengono modificati di conseguenza.
Unsupervised learning	K-means clustering	Utilizzando la similarità dei cluster di dati, l'algoritmo di clustering K-means (KM) definisce i cluster.
	Principal component analysis	L'analisi delle componenti principali (PCA) può fornire calcoli più rapidi e semplici, in quanto riduce la dimensione dei dati.
	Unsupervised neural network	La rete neurale non supervisionata (UNN) categorizza i dati in base alle loro somiglianze.

Tabella 1.1: Sintesi delle tipologie di ML e gli algoritmi correlati [5].

Supervised learning

Il supervised learning si basa sull'utilizzo di dati etichettati, dove a ciascun input è associato un output atteso. Durante la fase di addestramento, l'algoritmo apprende a riconoscere pattern all'interno del dataset e a sviluppare un modello predittivo in grado di generalizzare il comportamento osservato sui dati futuri. Questo processo richiede la supervisione umana, in quanto l'operatore fornisce i dati e monitora le prestazioni dell'algoritmo [6]. Il supervised learning è ampiamente utilizzato per problemi di classificazione e regressione, dove l'obiettivo è prevedere una variabile target nota.

Unsupervised learning

Al contrario, l'unsupervised learning opera su dati non etichettati, dove l'algoritmo deve individuare autonomamente schemi e strutture nascoste nel dataset. Questo approccio è

utile quando i dati non contengono una variabile target nota, e l'obiettivo è scoprire gruppi o pattern. Le applicazioni più comuni di questa tecnica sono il clustering e la riduzione dimensionale (es. K-means e PCA) [7]. L'unsupervised learning è spesso utilizzato per l'analisi esplorativa dei dati e il riconoscimento di anomalie.

Tecniche comuni di machine learning

A valle dell'introduzione delle categorie principali del machine learning, è importante esaminare alcune delle tecniche più diffuse rientranti in queste categorie e ampiamente utilizzate nelle applicazioni aziendali.

Differenti tipologie comunemente utilizzate sono i *decision tree* e le *random forest*. Gli alberi delle decisioni sono grafi con due tipologie di nodi: i nodi di probabilità e i nodi decisori. I primi mostrano le alternative a disposizione, i secondi compiono una scelta.
Le random forest, invece, combinano più decision tree per gestire database di notevoli dimensioni e migliorare la precisione del modello, riducendo l'overfitting, ovvero l'eccessiva aderenza al dataset di addestramento.

Una tecnica innovativa di recente diffusione sono le *suppor vector machines (SVM)*. Differentemente dalle reti neurali, le SVM si basano sul principio della minimizzazione del rischio strutturale, proiettando i dati in uno spazio multidimensionale e minimizzando l'errore marginale per la regressione. Il vantaggio derivante dall'utilizzo di questo approccio è l'assenza di minimi locali, a differenza delle reti neurali, l'algoritmo convergerà verso il minimo globale [6].

Le *reti neurali* sono una delle tecniche attualmente più utilizzate per implementare il ML. Ispirate al funzionamento del cervello umano, si basano sulla retropropagazione degli errori, secondo cui ciascun neurone riceve in input la somma pesata dell'output dei neuroni ad esso connessi. Tale tecnica assume che le reti siano costituite da più strati o layer: il layer di input, uno o più layer nascosti e il layer di output. La complessità della rete aumenta all'aumentare del numero di strati, così come la sua capacità computazionale. Una casistica particolare di reti neurali adatte ad operare con serie temporali sono le *recurrent neural network*, che lavorano tramite la back-propagation nel tempo, per cui esclusivamente l'output dei layer nascosti rappresenta l'input dei neuroni dello stesso strato.

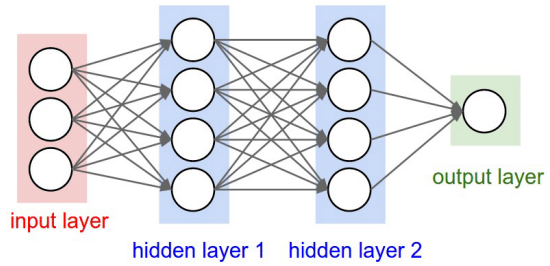

Figura 1.2: Rete neurale tradizionale.

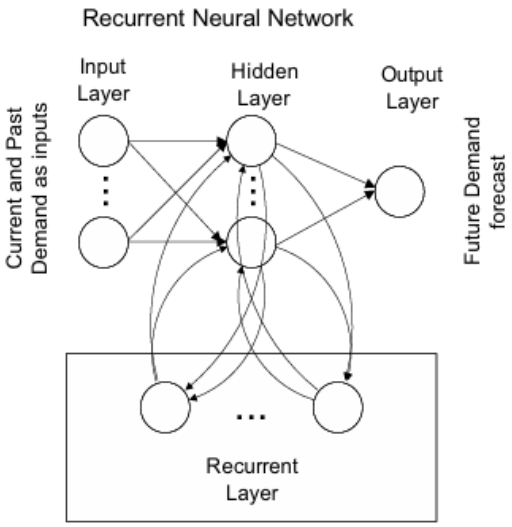

Figura 1.3: Reti neurali ricorrenti.

1.1.2 Deep learning

Le tecniche di machine learning tradizionali presentano limitazioni significative nella capacità di processare dati grezzi. Per anni, la realizzazione di sistemi di ML richiedeva particolare attenzione ingegneristica e notevole expertise nella creazione di strutture in grado di convertire il set di dati grezzi in una rappresentazione che permettesse, allo strumento, di identificare dei pattern o degli schemi interni all'input [8]. I metodi di *deep learning (DL)* si distinguono per la loro capacità di apprendere rappresentazioni su più livelli, attraverso la composizione di più moduli non lineari, ciascuno responsabile di un singolo livello di rappresentazione. Aumentando il numero di moduli, i modelli possono apprendere funzioni estremamente più complesse, amplificando aspetti rilevanti dell'input necessari per la classificazione, ma non visibili alle tecniche più semplici. Un aspetto chiave di questa metodologia è che i layer non vengono progettati dall'essere umano: vengono appresi automaticamente dai dati utilizzando delle procedure di apprendimento general-purpose, come la *back-propagation* [8].
Uno dei principali vantaggi delle deep neural networks rispetto alle reti tradizionali è la loro capacità di generalizzare nuove combinazioni di valori delle caratteristiche apprese. Inoltre, la composizione di layer di rappresentazione permette di strutturare i dati in input in modo tale da facilitare la previsione dell'output target. Un altro vantaggio rilevante è la capacità di gestire grandi quantità di dati e applicarsi a contesti complessi, come il riconoscimento delle immagini e l'elaborazione del linguaggio naturale. In questi ambiti, le reti profonde sono in grado di scoprire automaticamente le caratteristiche più rilevanti [9].

Le applicazioni pratiche del deep learning sono numerose e abbracciano molti settori. Ad esempio, il DL è alla base di tecnologie avanzate come il riconoscimento facciale, la guida

autonoma e la diagnosi medica tramite immagini.

Nonostante le potenzialità del deep learning, ci sono alcune sfide che devono essere affrontate. Il DL richiede grandi quantità di dati per l'addestramento, il che può rendere difficile l'applicazione in contesti dove i dati sono limitati. Inoltre, il training di modelli profondi è computazionalmente intensivo, richiedendo l'uso di hardware avanzato come GPU e TPU per accelerare il processo. Un'altra sfida rilevante è il cosiddetto "black box problem": spesso è difficile interpretare e spiegare le decisioni prese da una rete neurale profonda, il che può rappresentare un ostacolo in settori come la sanità, dove la trasparenza è cruciale [10].

Fortunatamente, l'evoluzione di algoritmi di ottimizzazione avanzati, come Adam e RMSProp, ha reso il DL più pratico e accessibile su larga scala, migliorando l'efficienza dell'apprendimento e riducendo i tempi di addestramento [11]. Inoltre, l'adozione di framework di deep learning come TensorFlow, PyTorch e Keras ha semplificato lo sviluppo e la sperimentazione di modelli, consentendo di sfruttare al meglio le capacità computazionali e rendendo il deep learning accessibile anche ai non esperti.

1.2 Reinforcement learning

Una tecnica di recente diffusione, estremamente innovativa, di machine learning è il *reinforcement learning (RL)*, una metodologia estremamente differente da quelle viste in precedenza, secondo cui, un *agent* affronta un problema mediante una sequenza di interazioni di tipo trial-and-error con un ambiente dinamico [12]. Un modello di RL viene definito tramite i suoi elementi costitutivi: **spazio delle azioni**, **spazio delle osservazioni** e **funzione di reward**. L'agente interagisce con l'ambiente eseguendo una delle azioni predefinite ad ogni iterazione. La scelta dell'azione dipende dallo stato dell'ambiente ricevuto come input. Ciascuna azione altera le condizioni dell'ambiente in modo positivo o negativo. La funzione di reward segnala all'agente quanto una data azione sia stata vantaggiosa, con l'obiettivo finale di massimizzare la ricompensa totale accumulata nel tempo. L'apprendimento avviene tramite un processo sistematico di trial-and-error.
Un modo intuitivo per far chiarire la relazione tra agente e ambiente è il seguente dialogo esemplificativo [12]:

Ambiente: Ti trovi nello stato 65. Hai 4 possibili azioni.
Agente: Azione 2.
Ambiente: Hai ottenuto un reward pari a 7. Ora sei nello stato 15. Hai 2 possibili azioni.
Agente: Azione 1.
Ambiente: Hai ottenuto un reward pari a 5. Ora sei nello stato 44. Hai 5 possibili azioni.
Agente: ...

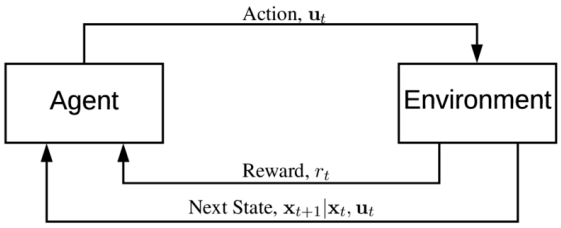

Figura 1.4: Schema concettuale del funzionamento di un algoritmo di reinforcement learning.

L'obiettivo dell'agente è trovare una *policy*, ossia la strategia attraverso la quale l'agente seleziona le azioni sulla base delle informazioni che ha a disposizione [13]. Tale policy

Il reinforcement learning differisce in vari aspetti dai più studiati algoritmi di apprendimento supervisionato, tuttavia, la principale risiede nel fatto che non vengono forniti input o output etichettati all'agente. Sarà lui stesso a determinare quale set di azioni è ottimale per massimizzare la reward complessiva [12]. Un ulteriore concetto chiave del RL è il trade-off tra exploration ed exploitation: per incrementare la reward un agente può scegliere di sfruttare (exploitation) azioni che in passato hanno generato una ricompensa elevata, ma per individuare nuove opportunità deve, in aggiunta, esplorare (exploration) azioni mai tentate fino a quel momento. L'agente deve, dunque, trovare un equilibrio tra le due strategie: provare un'ampia varietà di azioni, ripetendo quelle che portano a una reward maggiore [13].

L'utilizzo del reinforcement learning (RL) è stato a lungo limitato principalmente allo sviluppo di videogiochi, in cui l'agente rappresenta il personaggio principale che impara a interagire con un ambiente dinamico e a raggiungere determinati obiettivi. Nei videogiochi, il personaggio impara attraverso un processo di trial-and-error quali azioni portano alla massimizzazione della ricompensa, un processo che simula il funzionamento del RL nel mondo reale. Questo contesto offre un ambiente controllato in cui testare gli algoritmi di RL, prima di tradurli in applicazioni più complesse e dinamiche come la gestione della supply chain o l'ottimizzazione logistica. Tuttavia, il RL non è privo di limiti. Sebbene sia estremamente efficace in ambienti controllati, come i videogiochi, dove le regole sono ben definite e gli agenti hanno una chiara comprensione delle azioni possibili, risulta più complesso da applicare a contesti reali, come quelli industriali o logistici, in cui l'ambiente può essere molto variabile e imprevedibile. Gli algoritmi di RL devono affrontare sfide come la gestione di un numero elevato di variabili e l'aggiornamento continuo delle policy in tempo reale.

I benefici del reinforcement learning sono numerosi. Ad esempio, nel contesto della supply chain, il RL può essere utilizzato per ottimizzare le rotte di distribuzione o per migliorare

l'efficienza del picking in un magazzino. Un agente RL è in grado di apprendere quale strategia sia più efficace per massimizzare le prestazioni logistiche, adattandosi dinamicamente ai cambiamenti delle condizioni operative, come la disponibilità di risorse o le richieste dei clienti. Tuttavia, la scelta della policy rimane cruciale: trovare il giusto equilibrio tra l'esplorazione di nuove soluzioni e lo sfruttamento di quelle già apprese è fondamentale per garantire risultati ottimali nel lungo termine.

1.2.1 Deep reinforcement learning

L'avvento del deep learning ha avuto un impatto significativo in diverse aree del machine learning, portando a notevoli miglioramenti in diversi campi, come l'identificazione di oggetti e il riconoscimento vocale. Similarmente, il DL ha accelerato il progresso nelle tecniche di reinforcement learning, integrandosi con esse e dando origine al cosiddetto *deep reinforcement learning (DRL)*. Le reti neurali profonde consentono agli agenti RL di gestire input complessi e non strutturati, come immagini o sequenze temporali, che nei modelli tradizionali di RL risultano difficili da elaborare. La disponibilità di molteplici layer non lineari consente al DRL di apprendere rappresentazioni complesse e adattarsi a situazioni dinamiche in ambienti altamente variabili.
Il potenziale della combinazione di queste due metodologie è stato evidenziato dallo sviluppo di un algoritmo in grado di imparare a giocare a vari videogiochi Atari 2600 a un livello sovrumano. Gli algoritmi di DRL hanno trovato applicazione in svariati ambiti, dalla robotica, ai videogiochi, ma il vero driver dietro lo sviluppo di questi algoritmi è la visione di creare dei sistemi in grado di adattarsi al mondo reale, con infinite applicazioni possibili [14]. Un esempio notevole è il successo di AlphaGo di DeepMind, che ha sconfitto i campioni umani del gioco del *Go*, utilizzando una combinazione di tecniche di RL e deep learning [15]. Anche nella supply chain, il DRL può essere applicato per ottimizzare processi complessi, come la gestione della produzione o l'allocazione dinamica delle risorse, superando le limitazioni degli approcci RL tradizionali, che non sono in grado di gestire input di dati così complessi, migliorando così l'efficienza e l'efficacia nelle decisioni logistiche e operative. Rispetto al RL tradizionale, il DRL offre vantaggi significativi in termini di scalabilità e capacità di affrontare problemi più complessi e di maggiore portata. Tuttavia, anche il DRL ha limiti: il suo utilizzo richiede risorse computazionali notevoli, e il processo di addestramento può essere lungo e costoso, soprattutto in scenari dove è necessario simulare ambienti complessi. Inoltre, l'interpretabilità dei modelli DRL è spesso ridotta rispetto agli algoritmi più semplici, rendendo difficile la comprensione delle decisioni prese dall'agente [14].

1.3 Principali applicazioni dell'intelligenza artificiale nella supply chain di un azienda

La supply chain è una rete con determinate caratteristiche, può essere più o meno ramificata, andando a includere fornitori, distributori, retailers, magazzini, ma non solo, le esigenze dei consumatori sono il principale driver per le decisioni aziendali riguardo la configurazione del network. Le necessità dei clienti portano l'azienda a compiere scelte diverse riguardo il posizionamento dei nodi all'interno della catena, dunque, la conoscenza dei dati riguardo i consumi assume, sempre di più, un'importanza strategica in fase di progettazione. Questo rende necessaria una stretta collaborazione e condivisione di dati tra tutti gli attori della supply chain, in modo da garantire visibilità sui bisogni dei consumatori anche ai fornitori più a monte. Numerosi fattori, esterni ed interni, però, impattano sulle relazioni all'interno di un network logistico, rendendo complessa la cooperazione tra gli attori: l'allineamento degli obiettivi aziendali, la gestione delle relazioni a lungo termine, la riluttanza alla condivisione di dati rilevanti, la competenza del personale di supporto e il sistema di incentivi a supporto della gestione della supply chain nella sua interezza [6].
L'implementazione di sistemi basati su una delle tecniche di IA viste in precedenza possono portare notevoli benefici in termini di miglioramento delle performance della catena e maggiore visibilità sulla domanda finale da parte degli attori più a monte. Di conseguenza, le aziende stanno sempre più spesso adottando strumenti di machine learning o reinforcement learning per ottimizzare i processi interni. Nella sezione seguente verranno analizzate nel dettaglio le applicazioni più diffuse di ML e RL all'interno della supply chain di un'azienda.

1.3.1 Principali applicazioni del machine learning alla supply chain

La gestione della supply chain presenta sfide significative legate all'incertezza della domanda e alla mancanza di collaborazione tra gli attori coinvolti.
L'incertezza caratterizzante la domanda e la fornitura determina una distonia tra l'andamento del mercato e ciò che viene effettivamente colto dai livelli più a monte della catena, generando il cosiddetto *effetto bullwhip*, un fenomeno per cui una minima variazione a valle si amplifica a monte secondo un effetto frusta. Una possibile soluzione per mitigare questo disallineamento tra la produzione e le reali esigenze del mercato è rappresentato dall'utilizzo di strumenti avanzati per la previsione della domanda, come quelli basati sul machine learning.

Il forecasting è una tecnica ampiamente diffusa, che permette alle aziende di prevedere

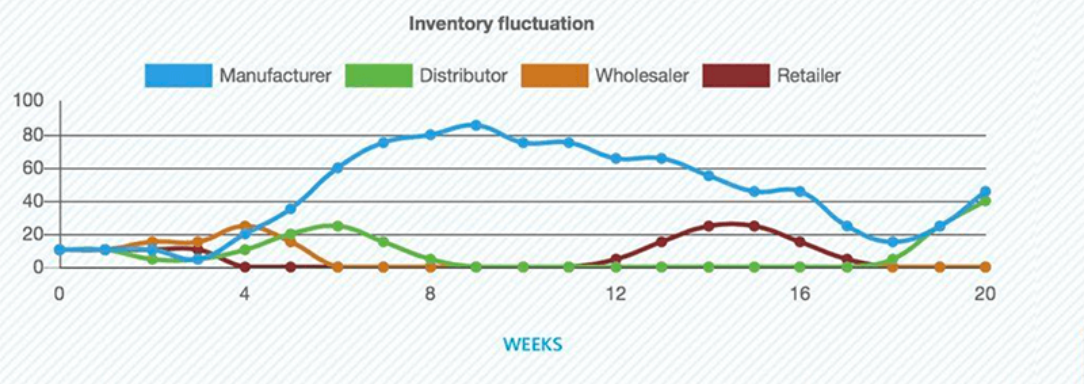

Figura 1.5: Fluttuazione del livello di inventario tra retailer e produttori.

la futura domanda di mercato, facilitando la pianificazione della fornitura e delle risorse necessarie. Tuttavia, una delle principali criticità legata all'utilizzo di queste metodologie è che, per definizione, le previsioni non sono accurate, con margini di errore che possono essere più o meno elevati, di conseguenza è necessario utilizzarle con particolare attenzione, tenendo conto di fattori che potrebbero influenzare i risultati. Il machine learning trova naturale applicazione in questo ambito, data la capacità di tali algoritmi di individuare pattern complessi all'interno di un set di dati. Tali algoritmi, dunque, rappresentano un'opportunità per ottenere performance migliori in termini di accuratezza delle previsioni. Inoltre, data la loro flessibilità, possono essere continuamente migliorati e sono adattabili alle variazioni dinamiche del mercato.

Il machine learning (ML) si distingue dalle tecniche tradizionali per la sua capacità di apprendere dai dati e adattarsi a contesti mutevoli, come quelli tipici della supply chain. Inoltre, consente di risolvere problemi non-lineari, per i quali, i metodi tradizionali, non sono in grado di convergere a una soluzione. Conseguentemente, questa tecnologia viene impiegata in numerose aree della catena logistica, dalla gestione dei fornitori, alla previsione della domanda.

La selezione e valutazione dei fornitori è una delle attività principali nella fase di acquisto. Tradizionalmente, tale processo viene svolto avvalendosi di un vendor rating system, classificando i fornitori, tenendo conto di vari fattori ritenuti rilevanti dall'azienda cliente. La criticità principale di questa tecnica, tuttavia, è rappresentata dalla sua staticità, una limitazione significativa in un contesto dinamico come quello attuale.
Le tecniche di ML permettono di superare i limiti dei metodi tradizionali, offrendo dinamicità e adattabilità, caratteristiche essenziali in un ambiente in costante cambiamento. In particolare, gli alberi decisionali e le support vector machines rappresentano la soluzione ideale per questa tipologia di problemi [5].

Un ulteriore possibile applicazione riguarda l'ambito della gestione del rischio all'interno della supply chain, un'area dove l'intelligenza artificiale, sino ad ora, ha trovato relativa-

mente poco spazio. Nonostante ciò, tecniche come i decision tree e le SVM potrebbero essere utilizzate per l'identificazione e classificazione dei rischi, mentre le reti neurali potrebbero supportare l'analisi quantitativa dei rischi individuati, con un approccio più accurato e dinamico [5].

L'utilizzo più diffuso delle tecniche di machine learning riguarda, tuttavia, la previsione della domanda futura di mercato. Come discusso in precedenza, l'obiettivo di tali strumenti è minimizzare l'effetto bullwhip, incrementando la visibilità sul mercato per gli attori più a monte della catena.

Uno studio portato avanti da *Carbonneau et al.*, pone a confronto i metodi di forecasting tradizionali con le previsioni realizzate da modelli di machine learning, evidenziando come questi ultimi portino a notevoli miglioramenti nell'accuratezza delle predizioni. L'efficacia dei vari modelli è stata valutata sulla base di dati forniti da *Statistics Canada*.

Tecniche di forecasting	**Testing set**		**Training set**	
	MAE	**Std. dev.**	**MAE**	**Std. dev.**
RNN	20.352	16.203	15.521	12.334
LS-SVM	20.485	17.304	3.665	3.722
MLR	21.396	19.705	15.007	15.041
NN	25.260	19.733	12.855	12.057
Moving average	25.481	19.253	18.205	13.028
Trend	27.323	24.198	17.995	17.292
Naïve	32.591	23.485	20.263	17.380

Tabella 1.2: Confronto tra le performance delle tecniche di forecasting sul data set [16].

La tabella 1.2 mette in luce i risultati dell'analisi svolta, confrontando le prestazioni dei vari modelli sulla base del *mean average error (MAE)*, una metrica che misura la differenza media tra i valori previsti e quelli effettivi. I risultati ottenuti mostrano le prestazioni superiori dei vari modelli di machine learning (recurrent neural networks (RNN), SVM, regressione multipla lineare (MLR) e neural networks (NN)) rispetto ad approcci più semplici.
L'accuratezza delle varie tecniche è stata valutata sulla base dei p-values derivati dai test-t, che indicano la probabilità che i risultati osservati siano dovuti al caso. I test confermano come le RNN e le SVM siano gli strumenti con il più elevato livello di accuratezza, grazie alla loro capacità di gestire dati complessi e non lineari, adattandosi dinamicamente ai cambiamenti nei pattern di dati [16].

1.3.2 Principali applicazioni del reinforcement learning alla supply chain

Gli algoritmi di reinforcement learning trovano un ampio range di applicazioni all'interno della supply chain, in particolare a supporto del processo decisionale. L'attenzione viene posta sui principali driver di performance delle catene logistiche. Nel corso degli anni si sono delineati sei fattori chiave di influenza sulle prestazioni di un network logistico [17]: i magazzini, la gestione del materiale a scorta, i trasporti, il flusso informativo, il sourcing e la definizione del prezzo.

Gestione del materiale a scorta

La gestione dell'inventory è un noto problema di ottimizzazione della supply chain, che implica decisioni riguardo il processo di riordino e il livello di stock da mantenere per non incorrere in stock out, generando un impatto su materie prime, semilavorati e prodotti finiti. I modelli tradizionali come l'*economic order quantity (EOQ)* offrono soluzioni rapide e semplici, richiedendo, tuttavia, alcune assunzioni: domanda costante nel tempo e assenza di asimmetria informativa. Approcci innovativi basati su RL possono oltrepassare questi limiti, data la loro capacità di apprendimento dinamica e di adattamento a contesti diversi.
La letteratura propone diverse soluzioni a tale problema mediante l'utilizzo di RL: lo sviluppo di un agente che si occupa del processo di procurement, vendita e pianificazione della produzione all'interno di una supply chain simulata [18]; l'introduzione del RL all'interno di una catena logistica ospedaliera per prevenire lo shortage di farmaci [19]; e infine, l'utilizzo del reinforcement learning per la risoluzione di un vendor-managed inventory (VMI) nel settore dei semiconduttori [20]. Questi sono solo alcuni esempi di come il problema dell'inventory management possa essere affrontato con un algoritmo estremamente versatile e potente come il RL [21].

Trasporto

Il trasporto comprende le attività che permettono la movimentazione dei prodotti da un sito della supply chain a un altro. L'ottimizzazione di questo processo richiede, molto spesso, l'identificazione dei percorsi più brevi rispettando vincoli come la capacità dei mezzi e i tempi di consegna accettati dal cliente. Tradizionalmente, tali problemi venivano risolti mediante l'uso di euristiche o metaeuristiche. Tuttavia, il reinforcement learning si è dimostrato in grado di ottenere risultati migliori con tempi di calcolo inferiori, grazie alla sua capacità di apprendere dalle interazioni con l'ambiente e adattarsi dinamicamente a cambiamenti nei vincoli operativi.
la letteratura propone diverse soluzioni basate sull'uso del RL: ad esempio, l'analisi del

vehicle routing problem, considerando last-mile operations a Bogotà, addestrando un agente a individuare la rotta più breve [22]; l'applicazione del reinforcement learning alla gestione del traffico ferroviario [23]; e infine, l'utilizzo del reinforcement learning per la gestione dei plotoni, giungendo a soluzioni migliori rispetto ai metodi tradizionali [24]. Questi esempi mostrano che il trasporto è un ulteriore ambito in cui il RL trova spazio, portando a un incremento delle prestazioni rispetto ai metodi tradizionali attualmente diffusi [21].

Flusso informativo

Il flusso informativo comprende tutte quelle attività che permettono di convertire dati grezzi in informazioni a supporto delle decisioni. Tipicamente, questo ambito è stato dominato delle applicazioni di machine learning, rendendo relativamente raro l'impiego del reinforcement learning, con pochi use cases presenti in letteratura. Tuttavia, esistono alcuni modelli interessanti che evidenziano il potenziale del RL in questo contesto: ad esempio, l'utilizzo del RL per la selezione della tecnica di forecasting più adatta; l'identificazione delle location di prodotti mancanti tramite il loro percorso pianificato e l'uso della tecnologia RFID [25]; l'addestramento di un agente per la risoluzione di conflitti tra attori della catena logistica [26]; e infine, lo sviluppo di un modello in grado di supportare il processo decisionale all'interno di un team, attribuendo pesi differenti alle varie proposte per giungere alla decisione ottimale [27]. Questi esempi dimostrano come, per quanto sia limitata, l'applicazione di questa metodologia all'ottimizzazione del flusso informativo genera benefici significativi, come il miglioramento della precisione nelle decisioni e l'ottimizzazione della gestione dei conflitti tra attori della supply chain [21].

Sourcing

Il processo di sourcing coinvolge attività chiave come la selezione e valutazione dei fornitori, oltre alle scelte di esternalizzazione. La letteratura raccoglie modelli realizzati allo scopo di supportare decisioni tattiche e strategiche riguardo la scelta dei supplier e le strategie di approvvigionamento: tra i vari esempi, si può citare lo sviluppo di un modello multi-agent che simula una gara di fornitura [28], la realizzazione di un sistema che suddivide gli ordini tra due fornitori complementari [29] e la simulazione delle decisioni di make-or-buy [30]. Nonostante l'applicazione del reinforcement learning nel sourcing sia ancora limitata, questi esempi dimostrano che l'uso di questa metodologia può generare benefici significativi, come il miglioramento della precisione nelle decisioni e l'ottimizzazione della gestione dei fornitori all'interno della supply chain [21].

Definizione del prezzo

La definizione del prezzo comporta le decisioni riguardo costi e strategie di prezzo, come la gestione delle fasi di gara, contrattazione, strategie di marketing e pianificazione delle vendite [17]. La letteratura tratta due applicazioni nel dettaglio: il *trading* e le *simulazioni di mercato*. tra gli esempi rilevanti, si possono citare l'addestramento di agenti per lo scambio di azioni su mercati simulati [31], l'uso di agenti per la gestione degli ordini cliente in termini di costi [32], la simulazione di diversi scenari di prezzo per valutare la risposta dei consumatori [33] e la valutazione del comportamento umano nel mercato, con particolare attenzione alla sensibilità al prezzo, simulata attraverso agenti basati su RL [34]. Tale ambito, dunque, è ricco di applicazioni che studiano il comportamento del mercato per accumulare informazioni utili a prendere delle decisioni più accurate, tenendo conto della risposta dei consumatori a variazioni di prezzo e migliorando così l'efficacia delle strategie di pricing.

1.3.3 Principali trend e innovazioni tecnologiche all'interno della supply chain

La crescente rilevanza strategica dell'IA viene, inoltre, messa in luca da uno studio pubblicato da Gartner, dove vengono discussi i vari trend nell'innovazione tecnologica della supply chain, mostrando come introdurre questi strumenti all'interno dei vari processi e i benefici che ne derivano [2]:

1. Cyber extortion: i leader tecnologici della supply chain dovrebbero attivare collaborazioni con la leadership IT per confermare che gli attacchi cyber vengano inclusi nei processi di gestione del rischio aziendale e sviluppare un playbook dettagliato di risposta agli incidenti, qualora si verifichino.

2. Supply chain data governance: l'emergere di potenti strumenti di analytics e di tecniche basate su intelligenza artificiale sta aumentando in modo massiccio le capacità di visibilità interfunzionale, di modellazione degli scenari e di automazione delle decisioni, diventa, dunque, una mission aziendale il mantenenimento di un elevato livello di qualità dei dati e un rigoroso processo di governance.

3. End-to-end sustainable supply chain: la legislazione in materia di sostenibilità sta crescendo a livello globale e sta determinando un passaggio dalla conformità volontaria a quella normativa. Di conseguenza, è necessario procedere al passaggio dalla semplice elaborazione di KPI, all'uso di tali indicatori come driver nel processo di decision-making.

4. Sistemi di visione abilitati all'intelligenza artificiale: soluzioni innovative di iperautomazione che combinano telecamere 3D industriali, software di computer vision

e tecnologie avanzate di riconoscimento di modelli basati su IA. Queste soluzioni sono in grado di acquisire, interpretare e trarre conclusioni autonomamente dalle immagini che i sistemi di visione vedono in tempo reale.

5. Augmented Connected Workforce (ACWF): le iniziative di ACWF riducono il tempo necessario dopo l'onboarding affinché un dipendente diventi pienamente produttivo e migliori il suo processo decisionale, è una strategia per ottimizzare il valore derivato da un lavoratore umano stabilendo un tessuto connettivo che ottimizza l'uso di IA, analisi della forza lavoro e aumento delle competenze.

6. Composite AI: si tratta dell'applicazione combinata di più tecniche di IA per migliorare l'efficienza e l'accuratezza dell'apprendimento, per ampliare il livello di rappresentazione delle conoscenze e, in ultima analisi, per risolvere una serie di problemi aziendali con l'obiettivo di ottenere miglioramenti delle prestazioni della supply chain.

7. Humanoid working robot di nuova generazione: combinano la consapevolezza sensoriale con la manipolazione mobile e la locomozione dinamica per eseguire lavori produttivi che in precedenza erano relegati agli esseri umani biologici.

8. Machine customers: attori economici non umani che ottengono autonomamente beni o servizi in cambio di un pagamento. Gli esempi includono: dispositivi o beni connessi all'IoT che effettuano ordini indipendentemente da un comando umano, algoritmi di rifornimento intelligenti che mantengono la disponibilità di materiali di consumo e assistenti intelligenti che suggeriscono offerte ai consumatori.

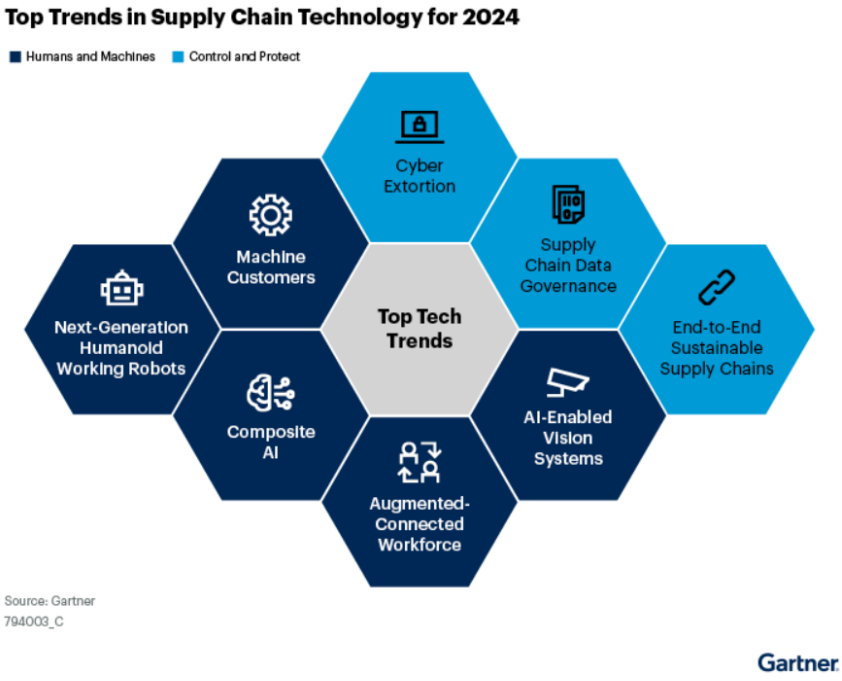

Figura 1.6: Principali trend nell'innovazione tecnologica della supply chain.

Lo studio in esame evidenzia le illimitate potenzialità dell'intelligenza artificiali e il ruolo chiave che potrebbe assumere all'interno della logistica, data la sua versatilità e adattabilità.

Come emerge dai tren discussi, il supply chain management è una disciplina estremamente complessa, che racchiude in sè numerose funzioni, coinvolgendo decisioni strategiche in differenti aree operative, che possono determinare la configurazione di un network logistico più o meno ramificato, in funzione degli obiettivi aziendali. L'intelligenza artificiale, in un contesto così articolato, trova una naturale applicazione, data la sua versatilità e capacità di fornire soluzioni specifiche per ogni esigenza. Esistono, infatti, diverse declinazioni di IA, che rispondono a diverse necessità all'interno di un ecosistema complesso come quello della catena logistica di un'azienda.

Capitolo 2

Warehouse management come fattore strategico nel miglioramento della supply chain

I magazzini rappresentano un elemento chiave all'interno del network logistico di un'azienda. Esistono diverse tipologie di centri di stoccaggio, classificabili a seconda della loro funzione:

- Depositi di fabbrica: destinati allo stoccaggio di materie prime, semilavorati o prodotti finiti.
- Depositi distributivi: possono essere centrali, periferici, dei centri distributivi (CE.DI.) o piattaforme per il cross-docking.

I magazzini, inoltre, possono essere di vario tipo a seconda dei materiali che devono contenere, influenzando la conformazione della struttura e le modalità di svolgimento delle operazioni al suo interno: ad esempio i cantilever, progettati per lo stoccaggio di carichi, si differenziano notevolmente dai magazzini destinati ai colli piccoli. Una categoria particolare di magazzini è quella destinata al *cross-docking*, una modalità operativa che non prevede il mantenimento del materiale all'interno della struttura per lunghi periodi. In queste tipologie di impianti, le merci scaricate vengono rapidamente organizzate e preparate per la spedizione, minimizzando il tempo di permanenza all'interno. Nonostante tali differenze, tutti i magazzini contribuiscono al funzionamento dell'azienda in modo analogo e con obiettivi comuni [35]:

- Ottimizzare i costi di trasporto (es., spedizione combinata, carico completo).
- Ottimizzare i costi di produzione (es., politica di produzione make-to-stock).
- Sfruttare sconti per acquisti in volume o a lungo termine.

- Supportare le politiche di servizio al cliente.
- Rispondere alle mutevoli condizioni di mercato (es., stagionalità, fluttuazioni della domanda, concorrenza), concorrenza).
- Superare le differenze di tempo e di spazio che esistono tra produttori e clienti.
- Realizzare una logistica al costo totale minimo, commisurata al livello di servizio desiderato.
- Sostenere i programmi just-in-time di fornitori e clienti.
- Consolidare diversi prodotti per ordine, fornendo un mix anziché un singolo prodotto.
- Offrire stoccaggio temporaneo di materiale da smaltire o riciclare (logistica inversa).
- Funzionare come area per le spedizioni (consegna diretta, cross-docking).

Quelle elencate sono le principali funzioni svolte da un magazzino, anche se esistono situazioni (come nel lean management, inventario virtuale, cross-docking) in cui le attività interne vengono ridotte. Tuttavia, in generale, le strutture per l'immagazzinamento di materiale restano essenziali all'interno di una catena logistica e svolgono un ruolo cruciale per garantire un buon funzionamento di una supply chain.

2.1 Processi interni a un magazzino

Le principali aree funzionali e il relativo flusso di materiale all'interno di un magazzino sono rappresentate nella figura 2.1, che illustra i processi principali svolti all'interno di un centro di stoccaggio [36]:

- *Ricezione del materiale*
- *Trasferimento all'area di stoccaggio e put-away*
- *Picking*
- *Sorting*
- *Cross-docking* (non sempre)
- *Spedizione*

Il flusso di materiale all'interno del magazzino si articola sui seguenti passaggi [37]:

1. Ricezione dei materiali provenienti dai fornitori.
2. Stoccaggio di tali prodotti.

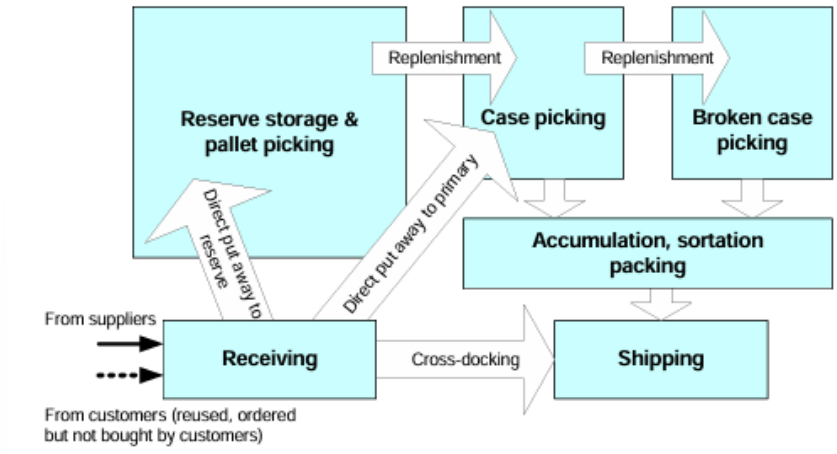

Figura 2.1: Principali aree funzionali e flusso di materiale all'interno di un magazzino [36].

3. Ricezione degli ordini dei clienti.
4. Fase di picking degli item richiesti.
5. Assemblaggio dell'ordine (Sorting).
6. Spedizione dell'ordine al cliente.

Un processo così articolato e complesso è caratterizzato da varie criticità relative alla disponibilità di risorse e di spazio, alla cooperazione tra diverse funzioni interne, dunque, è necessario compiere varie decisioni volte a ottimizzare le attività di magazzino, garantendo un corretto funzionamento della struttura.

Ricezione e spedizione del materiale

L'arrivo e la spedizione dei prodotti sono attività che avvengono tramite un *carrier*, responsabile anche del carico e dello scarico degli articoli presso i dock dedicati. Tale processo implica la necessità di prendere decisioni riguardo l'assegnazione delle baie ai vari carrier in arrivo, la pianificazione degli arrivi e l'allocazione e l'utilizzo delle risorse a disposizione. Il processo decisionale è influenzato da vari fattori e deve sottostare ad alcuni vincoli, tra cui il tempo di ciclo necessario per l'attività di carico/scarico dei mezzi, il layout delle baie e dell'area di stoccaggio, le politiche di gestione (es. un cliente per dock), i requisiti di throughput di ciascuna baia e la disponibilità di informazioni sulle spedizioni in entrata e in uscita dal magazzino.
L'introduzione di tecnologie innovative come RFID, GPS e advanced shipping notices (ASN) rende possibile una maggiore visibilità e controllo sugli arrivi e le partenze di materiale, agevolando le attività di receiving e shipping della merce. Tali tecnologie permettono di ottimizzare il flusso delle merci, ridurre i tempi di attesa e migliorare l'accuratezza delle operazioni.

La letteratura si è focalizzata principalmente su problemi di assegnazione *carrier-to-dock*. Alcuni studi rilevanti sono includono: lo sviluppo di un modello bilineare per l'assegnazione dei mezzi alle baie di carico/scarico [38]; la realizzazione di un tool per l'attribuzione ottimale dei mezzi alle rispettive baie, con l'obiettivo di minimizzare i costi operativi [39]; l'elaborazione di uno strumento che risolva il problema di assegnazione, minimizzando il tempo di percorrenza e il tempo di attesa in coda [40].

Stoccaggio dei prodotti

L'attività di stoccaggio coinvolge tre decisioni fondamentali: la quantità di materiale da tenere a scorta, la frequenza di riordino e il posizionamento dei vari codici all'interno del magazzino. Le prime due riguardano le scelte di lot sizing e le politiche di riordino, che avranno un impatto sulla supply chain nella sua interezza, dalle relazioni con i fornitori alla configurazione del centro di stoccaggio, considerando le modalità di trasporto e la frequenza delle spedizioni.
Una delle metodologie più diffuse è l'*economic order quantity (EOQ)*, che stabilisce la dimensione ottimale del lotto in grado di minimizzare i costi logistici totali (costi di emissione dell'ordine, costo di mantenimento a magazzino e costo di acquisto).
La terza decisione, relativa alla posizione ottimale dei vari codici all'interno del magazzino, richiede un'analisi più dettagliata, che verrà trattata in una sezione successiva. In questa parte, verranno esplorate le diverse strategie di assegnazione delle locazioni, valutando come ciascuna possa influenzare le performance operative del magazzino.

$$EOQ = \sqrt{\frac{2DA}{h}}$$

dove:

- D è la domanda media annua del prodotto.
- A è il costo di emissione dell'ordine.
- h è il costo di mantenimento del prodotto a magazzino all'anno.

Il valore ottenuto rappresenta la dimensione ottimale del lotto, in corrispondenza della quale si trova il minimo della curva del costo totale.
Tipicamente, questa tecnica viene associata alla politica di riordino del *reorder point (ROP)*. Tale strategia prevede di effettuare un ordine di fornitura quando il livello di scorta raggiunge un valore predefinito R. Questo valore viene calcolato tenendo conto del consumo medio giornaliero e del lead time. Il punto di riordino è generalmente maggiore della sola domanda durante il lead time, in quanto include una scorta di sicurezza per prevenire stock-out causati da variazioni della domanda o ritardi nelle consegne.

Sorting

L'attività di sorting diventa necessaria quando, in fase di picking, vengono prelevati prodotti per più ordini contemporaneamente. Può essere realizzata durante la fase di picking (*sort-while-pick*), oppure a seguito di quest'ultima (*sort-after-pick*). I criteri di raggruppamento possono variare, ma tipicamente includono il cliente, la destinazione o la tipologia di prodotto, con l'obiettivo di ottimizzare le spedizioni e massimizzare i viaggi a pieno carico.
Il sorting può essere eseguito manualmente da parte degli operatori oppure con sistemi di smistamento automatizzati (*conveyor belt system*), progettati per indirizzare i prodotti verso i punti di spedizione corretti, grazie all'identificazione dell'articolo tramite codici a barre o RFID. Nei magazzini più avanzati, l'uso di robot mobili autonomi (AMR) permette un'ulteriore automazione e ottimizzazione delle operazioni di sorting, migliorando l'efficienza e riducendo i tempi di elaborazione degli ordini.

Figura 2.2: Esempio di un conveyor belt system di Dematic.

2.2 Picking: inquadramento e analisi del problema

Il processo di picking è da sempre considerato l'attivita più dispendiosa in termini di ore lavoro e di costi per la gran parte dei magazzini. Si stima, infatti, che impatti per circa il 55% sulle spese operative complessive. Conseguentemente, le aziende che mirano a ridurre i costi e incrementare la produttività all'interno dei loro magazzini o centri distributivi pongono particolare attenzione sull'ottimizzazione del processo di prelievo.
Il picking viene definito come il processo di raccolta dei prodotti dalle loro posizioni di stoccaggio, in risposta a una specifica richiesta del cliente [41]. Si tratta di un'attività chiave per il miglioramento delle performance di un magazzino e, di conseguenza, è considerato uno degli ambiti a più alta priorità per ottenere incrementi di produttività.

Recenti tendenze in ambito produttivo e distributivo hanno rivoluzionato il modo di gestire e progettare il prelievo degli articoli: si assiste a uno spostamento verso lotti di dimensioni ridotte, alla consegna diretta dei prodotti al punto di utilizzo, a una crescente personalizzazione degli articoli e alla riduzione delle finestre temporali tra l'arrivo di un articolo in magazzino e la sua spedizione al cliente. Inoltre, i magazzini di capacità contenuta vengono sostituiti da un numero minore di grandi centri di stoccaggio, allo scopo di realizzare economie di scala. L'importanza strategica dell'attività di picking deriva da un crescente numero di prodotti da prelevare, così come l'estensione della superficie da coprire, mentre la finestra temporale per il completamento dell'operazione rimane invariata [41]. Infatti, dal momento in cui un ordine giunge al centro di stoccaggio sino al momento in cui arriva a destinazione, esistono diverse opportunità di errore in termini di accuratezza e completezza.

Le problematiche caratterizzanti il processo di prelievo generano un impatto non trascurabile sull'efficienza e l'efficacia di una supply chain, dilatando i tempi di consegna e incrementando le probabilità di errore umano durante il flusso di materiale attraverso un magazzino. Esistono, tuttavia, alcune tecniche volte a ridurre tali impatti, migliorando le prestazioni dei centri di stoccaggio e garantendo un flusso operativo più efficiente, con l'obiettivo di fornire al cliente il livello di servizio desiderato.

2.2.1 Warehouse management system: migliorare la visibilità e il controllo all'interno del magazzino

La visibilità all'interno di un magazzino è un tema centrale per incrementare le performance di un centro distributivo e garantire maggiore precisione nell'allestimento degli ordini e nel processo di prelievo. La conoscenza delle locazioni dei prodotti, degli ordini in arrivo per ogni giornata e di altre informazioni rilevanti sono essenziali per assicurare un corretto funzionamento di un sito di stoccaggio e facilitare il processo decisionale dei manager responsabili di tutte le attività di una struttura. Negli ultimi anni, dunque, si è osservata una rapida diffusione di sistemi che supportassero la gestione di un magazzino: i *warehouse management system (WMS)*. Secondo uno studio condotto da ELA/AT Kearney (2004), le attività di magazzino contribuiscono ai costi logistici per circa il 20%: si tratta, quindi, di processi chiave all'interno della supply chain di un'azienda. Inoltre, la sempre maggiore integrazione delle catene logistiche, la globalizzazione e il rapido sviluppo tecnologico che si sta osservano in questi anni, hanno posto i centri di stoccaggio al centro dell'attenzione dei vertici aziendali per la ricerca dell'efficienza. L'obiettivo di un WMS è quello di ridurre l'impatto di tali cambiamenti, adottando metodologie innovative nel monitoraggio della movimentazione e dello stoccaggio della merce e dei processi ad essi associati [42]. Un warehouse management system è un software che mira a migliorare l'efficienza dei magazzini, monitorando il materiale a disposizione, registrando tutte le

transazioni. La funzionalità primaria di un WMS è ricevere informazioni da un sistema di più alto livello e tradurle a livello operativo. Uno studio è stato condotto sulla più grande azienda operante nel settore del retail in India. Questa società gestisce oltre tre milioni di prodotti all'anno, con più di trenta supply chain da gestire in simultanea, ciascuna che richiede specifiche soluzioni logistiche. L'analisi è stata realizzata su tre magazzini, di cui soltanto uno dotato di WMS [42].

Processo	**Time Savings per Order (in mins)**	**Miglioramento nei processi (%)**
Receiving	159	68
Put-Away	14	36.84
Picking	49	77.78
Packaging	35	68.62
Dispatch	424	94.2

Tabella 2.1: Miglioramento nelle performance dei processi di magazzino a seguito dell'implementazione del WMS [42].

La tabella 2.1 mette in evidenza i le prestazioni ottenute a seguito dell'implementazione di un warehouse management system, confrontando i miglioramenti rispetto a una situazione in cui tale sistema era assente. I benefici risultano essere significativi, con tutte le attività che presentano performance decisamente superiori.

Metric	**Non WMS**	**WMS**
Appointment Scheduling	No	Yes
Dock Scheduling	No	Yes
Is ASN enabled scheduling?	No	Yes
Space Allocation	Random	Algorithm Based
FEFO concept	No	Yes
Pick Accuracy	Low	High
Picking Sequence	Random	Algorithm Based
Cluster Pick	No	Yes
Overall Time (mins)	773	236
Traceability	Poor	100% Tracking
Capacity (units/day)	6000/day	24000/day
Manpower	97	57

Tabella 2.2: Confronto tra situazioni con e senza WMS [42].

La tabella 2.2 si pone l'obiettivo di valutare i benefici intangibili derivanti dall'implementazione di un warehouse management system, quali la possibilità di pianificare l'arrivo dei mezzi e la prenotazione delle baie di carico, il supporto nel processo di put-away, la tracciabilità della merce, l'aggiornamento automatico delle informazioni e molti altri vantaggi operativi.

L'ottimizzazione delle attività di picking rappresenta una priorità strategica per molte aziende, poiché è una delle operazioni più onerose in termini di costi e tempo. Come evidenziato, l'implementazione di strumenti di gestione più avanzati, come i warehouse management system, può apportare significativi miglioramenti nelle prestazioni operative, riducendo le inefficienze e migliorando la precisione dei processi.

2.2.2 Storage assignment problem: determinare le locazioni ottimali per accelerare il picking

Lo *storage assignment problem (SAP)* si occupa di individuare le locazione ottimali per ciascun prodotto, allo scopo di minimizzare il tempo di picking e di massimizzare l'utilizzo del magazzino, rispettando vincoli relativi alla capacità di stoccaggio, alla disponibilità e produttività degli operatori e alle politiche di spedizione [37]. Esistono varie policy per l'allocazione degli spazi di stoccaggio, tra cui le più rilevanti sono: l'*allocazione casuale*,

l'*allocazione nella closest-open location*, l'*allocazione dedicata*, il *full turnover storage* e l'*allocazione class-based*.
La random storage prevede l'attribuzione casuale di una location, selezionata tra tutte quelle disponibili con la stessa probabilità. Sebbene questa strategia consenta un'elevata utilizzazione dello spazio, incrementa la distanza percorsa dall'operatore in fase di picking. Una strategia derivata dall'assegnazione casuale è la closest-open location storage policy. Implementare questa politica di storage assignment significa attribuire a ciascun prodotto la prima locazione disponibile incontrata dall'operatore, riducendo il tempo di allocazione, ma senza ottimizzare necessariamente il percorso.
Il dedicated storage, invece, attribuisce a ciascun prodotto una posizione fissa all'interno del magazzino. Questo comporta un uso meno efficiente degli spazi, in quanto alcune location possono rimanere vuote se gli articoli assegnati sono esauriti, oppure solo parzialmente occupate se tali codici non raggiungono il livello di scorta massimo. Tuttavia, questa strategia è ampiamente diffusa, in quanto permette agli operatori di familiarizzare con le locazioni dei prodotti, riducendo i tempi di prelievo. Inoltre, permette di posizionare gli articoli più pesanti alla base della scaffalatura, facilitando il picking.
La full-turnover policy distribuisce i prodotti sulla base del loro turnover: gli articoli alto-vendenti vengono posizionati nelle location più facilmente accessibili dagli operatori, viceversa, gli slow-movers, sono collocati nella sezioni posteriori del magazzino.
L'allocazione class-based combina alcune delle suddette tecniche. L'inventario viene suddiviso in classi, solitamente utilizzando il principio di Pareto: i fast-movers rappresentano circa il 20% del materiale totale. Tipicamente, vengono definite tre classi: la **classe A** per i fast-movers, la **classe B** per i prodotti intermedi e la **classe C** per gli slow-movers. Ogni classe così costituita viene assegnata a una sezione specifica del magazzino. Tuttavia, esperimenti di simulazione hanno dimostrato che, in termini di distanza percorsa, la full-turnover policy ottiene prestazioni superiori rispetto all'allocazione class-based. La differenza tra le due strategie dipende dalla tecnica di partizione e dal metodo di routing considerato. L'implementazione di un metodo class-based risulta, però, più semplice rispetto a un full turnover storage, poichè richiede una minore complessità nella gestione delle locazione e nel monitoraggio del turnover dei prodotti [43].

Le storage assignment policies descritte finora sono comunemente impiegate in gran parte delle aziende, ma non tengono conto delle possibili relazioniesistenti tra due o più prodotti. Infatti, molto spesso, i consumatori acquistano due o più prodotti insieme, di conseguenza potrebbero essere vantaggioso porli in locazioni adiacenti. Il posizionamento di articoli simili all'interno della stessa area di stoccaggio è noto come *family grouping* [41]. Questo metodo di assegnazione può essere combinato con una delle tecniche precedentemente analizzate.
La letteratura ha esaminato approfonditamente questo tema. Uno studio, in particolare,

ha trattato il tema dei requisiti di spazio per l'allocazione del materiale in un magazzino portuale, seguendo due policy diverse: random storage assignment policy e family grouping. I risultati hanno evidenziando una maggiore necessità di superficie disponibile nel caso del raggruppamento di prodotti rispetto all'assegnazione casuale [44].

Le politiche di assegnazione degli spazi e le strategie di picking sono strettamente interconnesse: la disposizione dei prodotti influisce direttamente sulla velocità e sulla precisione delle attività di prelievo. Nella sezione successiva verranno analizzate le principali strategie di picking, evidenziando le tecniche più utilizzate per ottimizzare questa fase critica della gestione del magazzino.

2.2.3 Strategie di picking più diffuse

Durante la progettazione di un sistema di picking, è fondamentale scegliere la strategia di prelievo che meglio si allinea con gli obiettivi di performance desiderati. La scelta richiede di tenere in considerazione numerosi fattori, tra cui il throughput, i costi operativi, la superficie da coprire, la tipologia di prodotti e il tempo di evasione degli ordini che si vuole mantenere. Le possibili alternative tra cui compiere una scelta includono le seguenti tecniche di picking [45]:

- *Discrete picking*: un operatore prelevi tutti gli item relativi a un singolo ordine durante un unico pick-tour. Questa tecnica è semplice da implementare, ma estremamente labor-intensive, poiché non ottimizza il numero di viaggi all'interno del magazzino.
- *Batch picking*: più ordini vengono raggruppati e un picker preleva tutti gli articoli relativi a un batch. È utile in contesti in cui gli ordini contengono articoli diversi tra loro e può ridurre il tempo di percorrenza totale.
- *Zone picking*: ciascun picker viene assegnato a una determinata area del magazzino ed è responsabile del prelievo esclusivamente in quella sezione. Questa tecnica è particolarmente efficace quando i prodotti sono organizzati in base alla frequenza di prelievo o ad altre caratteristiche logistiche.
- *Wave picking*: prevede che una determinata quantità di ordini venga prelevata in una finestra temporale limitata. Questa modalità può essere combinata con altre tecniche, come il batch picking o lo zone picking, per migliorare l'efficienza nel rispetto delle tempistiche di spedizione.

Non esiste, tra quelle proposte, una soluzione universalmente migliore, ciascuna di esse, infatti, risulta più efficace in contesti specifici. Ad esempio, il discrete picking è di semplice implementazione, ma è anche altamente intensivo in termini di manodopera. Il batch picking è consigliato quando gli ordini includono articoli eterogenei, mentre lo zone picking

può portare notevoli benefici se le locazioni sono state assegnate in modo ottimale per minimizzare i tempi di percorrenza. Infine, il wave picking risulta vantaggioso nei contesti in cui è essenziale rispettare tempistiche stringenti per la spedizione.

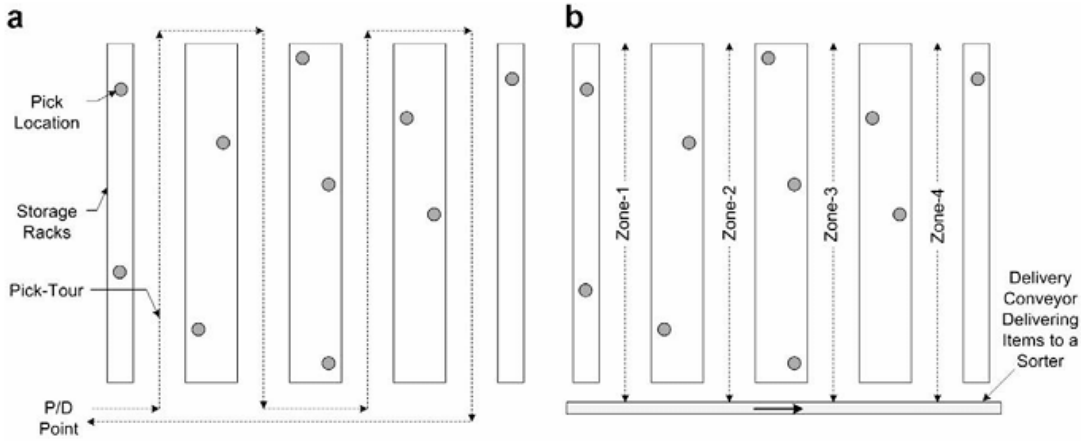

Figura 2.3: Esempi di batch (a) e zone (b) picking.

La letteratura ha ampiamente studiato questo ambito, focalizzandosi sul problema del batching, proponendo diversi approcci e modelli per ottimizzarne l'efficacia. Uno studio si concentra su valutare le performance di alcuni algoritmi di raggruppamento, considerando diverse storage assignment policies [46]; un altro contributo significativo presenta un'euristica che penalizza la tardiness e la earliness degli ordini [47]; un ulteriore trattazione propone un algoritmo che mira a minimizzare la somma pesata dei tempi di picking [48]; e infine, uno studio propone lo sviluppo di un modello per l'ottimizzazione del batching, formulando il problema del batching come un problema di partizionamento e risolvendolo con un algoritmo i branch-and-price, che si è rivelato efficace nel risolvere il problema in modo ottimale riducendo al minimo i tempi di picking e migliorando l'efficienza complessiva delle operazioni di magazzino. [49].

L'evoluzione tecnologica sta aprendo nuove possibilità nel campo del picking. L'introduzione di sistemi automatizzati e l'uso di intelligenza artificiale stanno rivoluzionando la gestione operativa, permettendo non solo una riduzione degli errori umani, ma anche l'ottimizzazione continua delle attività in tempo reale. Il futuro vedrà un'ulteriore integrazione di queste tecnologie, che renderanno le strategie tradizionali di picking parte di un sistema di gestione ancora più dinamico ed efficiente.

2.2.4 Tipologie di routing policies

Il trasporto dei prodotti e il tempo di percorrenza nel processo di picking rappresentano circa il 50% del tempo di evasione di un ordine [41], di conseguenza la scelta della strategia di routing svolge un ruolo cruciale nella progettazione di un sistema di prelievo efficiente.

La definizione di una route di picking mira a individuare la sequenza ottimale di locazioni da visitare al fine di minimizzare la distanza percorsa dall'operatore.
Esistono numerose politiche adottabili a seconda del layout del magazzino e delle esigenze operative. Le policies più comunemente utilizzate includono [50]:

- *S-shape*
- *Return*
- *Largest gap*
- *Midpoint*
- *Composite*

Le strategie più diffuse, come la S-shape e la largest gap, sono spesso preferite per la loro semplicità e comprensibilità da parte degli operatori. D'altra parte, le politiche come la midpoint e la returnsono note per i ridotti tassi di errore. La largest gap, in particolare, tende a essere la più utilizzata rispetto alla midpoint grazie alle sue prestazioni superiori in termini di efficienza. Infine, la composite policy, sebbene più complessa da implementare, è una delle euristiche che porta a performance migliori, combinando elementi di altre strategie per ottimizzare il percorso di prelievo.
La figura 2.3 illustra il funzionamento delle principali politiche di routing, che verranno analizzate nel dettaglio di seguito.

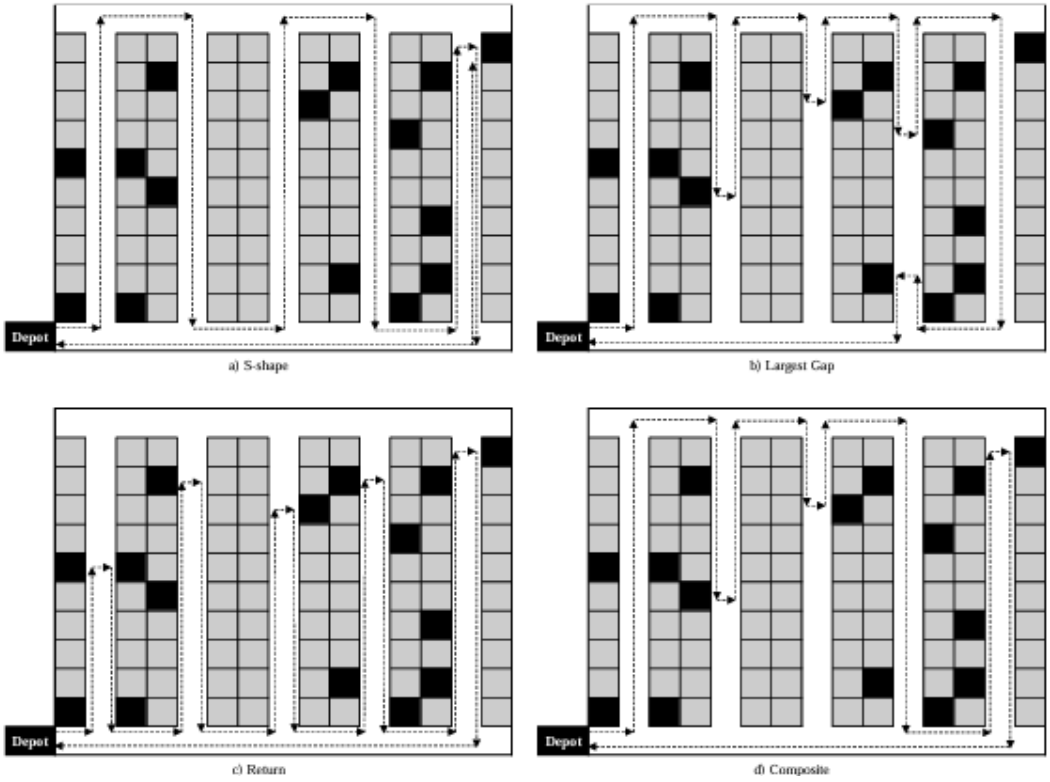

Figura 2.4: Principali picking routing policies.

S-shape routing policy

La S-shape routing policy è una delle politiche di routing più semplici: un operatore entra in un corridoio da un lato e, se è presente almeno un item da prelevare, lo attraversa completamente, uscendo dal lato opposto, procedendo al successivo. Numerosi studi hanno dimostrato che tale strategia ottiene performance migliore quando la densità di prelievo per corridoio è elevata. Tuttavia, se la lista di prelievo contiene pochi articoli, questa policy può risultare inefficiente [51] [52].

Largest gap routing policy

La largest gap routing policy pone a confronto le distanze tra l'item da prelevare e i lati del corridoio, selezionando la direzione con distanza minore. Il "gap" fa riferimento alla distanza tra due articoli consecutivi, o tra il primo prodotto e il lato frontale del corridoio, oppure tra l'ultimo articolo e la fine del corridoio. Il gap maggiore è quello che non verrà percorso dal picker. Tale strategia risulta particolarmente efficiente nei magazzini con lunghi corridoi e liste di prelievo contenute [53], ma diventa inefficace con un numero elevato di articoli da prelevare [51].

Return routing policy

La return routing policy è un'euristica che prevede l'entrata e l'uscita dell'operatore dallo stesso lato del corridoio, prelevando tutti gli item all'interno. Tale strategia raggiunge performance migliori in contesti con ordini di piccole dimensioni [52], ma tende a diventare meno efficiente all'aumentare del numero di prelievi da effettuare [53]. La return policy è particolarmente consigliata per magazzini con corridoi di lunghezza moderata, in modo da ridurre la distanza complessiva percorsa dall'operatore [54].

Composite routing policy

La composite routing policy combina caratteristiche delle euristiche s-shape e return, con l'obiettivo di minimizzare la distanza percorsa tra i punti di prelievo più distanti in due corridoi adiacenti. Dunque, a ogni corridoio, l'operatore deciderà da quale lato uscire in base alla distanza da percorrere, optando per un strategia di return o di S-shape. Studi dimostrano che le prestazioni raggiungibili da questa policy sono comparabili a quelle ottenibili con metodi ottimali, in particolar modo per liste di prelievo con un elevato numero di articoli [53].

La scelta della strategia di routing influisce in modo significativo sulle performance complessive del magazzino, determinando non solo il tempo necessario per completare il picking, ma anche la precisione e la riduzione degli errori durante il processo. Mentre politiche semplici come la S-shape e la return sono ideali per contesti operativi con un

basso numero di prelievi, strategie più sofisticate come la composite offrono risultati simili ai metodi ottimali in contesti con un elevato numero di ordini e corridoi più lunghi. La scelta della giusta routing policy dipende strettamente dalle caratteristiche del layout del magazzino, dal volume degli ordini e dalla densità di prelievo in ciascun corridoio. Come evidenziato dagli studi analizzati, non esiste una politica di routing che sia universalmente migliore, ma la combinazione e l'adattamento delle strategie possono portare a significativi miglioramenti nelle performance operative [53].

2.3 Soluzioni basate sull'IA per l'ottimizzazione del picking

L'intelligenza artificiale ha trovato ampio spazio di applicazione all'interno del processo di picking, un'attività ch ha un impatto significativo in termini di costi e impiego di risorse sulla supply chain. L'introduzione dell'IA, come discusso nel capitolo precedente, ha ha rivoluzionato diverse aree di un network logistico dalla selezione dei fornitori alla previsione della domanda, fino alla gestione ottimale dell'inventario.
Il picking, allo stesso modo, essendo una delle operazioni più impattanti sulle prestazioni di una catena logistica, ha visto notevoli miglioramenti grazie all'introduzione di tecnologie IA. L'IA, in particolare, ha reso possibile un'evasione degli ordini più rapida e accurata, con una riduzione significativa dei costi operativi, dal ricevimento della merce alla spedizione [55]. Uno dei principali contributi dell'IA in questo contesto è rappresentato dall'utilizzo di robot collaborativi, in particolare gli *autonomous mobile robots (AMR)*, che utilizzano algoritmi di machine learning per ottimizzare i percorsi di picking e ridurre al minimo le distanze percorse dagli operatori. Un ulteriore applicazione degna di nota riguarda attività strettamente connesse al picking, come il sorting e il packing [56]. Algoritmi di intelligenza artificiale sono in grado di analizzare la configurazione degli ordini e determinare la configurazione ottimale per il packaging, oltre a suggerire i materiali più adatti per l'imballaggio. Inoltre, sistemi di sorting automatizzati dotati di vision systems basati su IA consentono di classificare i prodotti in base a criteri predefiniti, migliorando ulteriormente l'efficienza operativa.

2.3.1 Ottimizzazione dei percorsi di picking tramite l'utilizzo di intelligenza artificiale

L'ottimizzazione dei percorsi di picking tramite l'uso dell'intelligenza artificiale rappresenta una delle principali innovazioni nella gestione del magazzino, poiché permette di migliorare l'efficienza operativa e ridurre i costi legati alle operazioni di prelievo. Grazie a tecniche avanzate come il deep reinforcement learning (DRL), è possibile superare i

limiti dei metodi tradizionali, che spesso si basano su euristiche statiche per determinare i percorsi degli operatori. Questi approcci convenzionali, come le strategie di routing S-shape e return, pur essendo ampiamente utilizzati, non sono sempre in grado di adattarsi rapidamente alle variazioni delle condizioni del magazzino o alle fluttuazioni della domanda, riducendo l'efficacia complessiva delle operazioni di prelievo.
L'impiego del DRL, al contrario, consente di sviluppare algoritmi che apprendono a ottimizzare i percorsi in modo autonomo, adattandosi dinamicamente a un ambiente in continua evoluzione. Questo processo di apprendimento si basa sulla capacità dell'algoritmo di esplorare diverse soluzioni, valutando le azioni sulla base di un sistema di ricompensa, e di scegliere i percorsi che minimizzano la distanza percorsa dai picker o dai robot, massimizzando la produttività complessiva.
Un altro aspetto rilevante dell'integrazione del DRL nell'ottimizzazione del picking è la sua capacità di gestire ambienti complessi e di lavorare in sinergia con sistemi di robotica, come gli autonomous mobile robots (AMR). Questi robot, combinati con algoritmi di IA avanzati, possono eseguire rapidamente i percorsi di prelievo calcolati, riducendo ulteriormente la necessità di intervento umano e migliorando la sicurezza complessiva delle operazioni. La flessibilità degli algoritmi di machine learning, inoltre, permette di migliorare costantemente le performance del sistema, poiché essi possono essere riaddestrati in modo continuo, adattandosi a cambiamenti come la redistribuzione delle merci all'interno del magazzino o variazioni nelle modalità di ordine da parte dei clienti. Tuttavia, l'implementazione del DRL per l'ottimizzazione dei percorsi di picking presenta anche alcune sfide. Una delle principali difficoltà risiede nella necessità di raccogliere una grande quantità di dati storici per addestrare i modelli in modo efficace. Questo può essere un ostacolo per le aziende che non dispongono di sistemi digitalizzati per la raccolta e l'analisi dei dati. Inoltre, i modelli di DRL richiedono tempi di addestramento considerevoli e risorse computazionali avanzate, che possono aumentare i costi iniziali di implementazione. Nonostante ciò, i vantaggi a lungo termine che derivano dall'utilizzo del DRL, come la maggiore efficienza operativa e la capacità di rispondere in modo rapido a cambiamenti nel flusso di lavoro, rendono questi strumenti altamente promettenti per le aziende che desiderano modernizzare la loro gestione del magazzino e migliorare la competitività [57].

2.3.2 Trend e sfide dell'IA nell'automatizzazione dei magazzini

Recenti trend nell'automatizzazione dei magazzini tramite l'uso di intelligenza artificale stanno radicalmente trasformando il mondo della logistica, introducendo tecnologie innovative per l'ottimizzazione dei processi, per il miglioramento del decision-making process e facilitando l'adattamento a un ambiente dinamico come quello del magazzino. Una delle innovazioni più rilevanti è l'*edge computing*, un paradigma tecnologico che consente

il proessamento dei dati in prossimità della loro fonte di origine, riducendo la necessità di trasmissione delle informazioni a server centrali e accelerando, di conseguenza, il processo decisionale. Questo approccio, in un contesto di magazzino, viene utilizzato su robot, sensori e lettori RFID, permettendo loro di operare in modo locale senza dover dipendere da connessioni costanti al cloud [58].
Gli algoritmi di IA integrati nei dispositivi edge, sono in grado di analizzare i dati provenienti dai sensori, identificare schemi e agire in autonomia, senza dover attendere istruzioni dai server centrali. Ciò consente ai magazzini di rispondere rapidamente a cambiamenti nei livelli di scorte, all'andamento della domanda e alle condizioni operative [59].

Figura 2.5: Schema di funzionamento dell'edge computing.

Un'ulteriore applicazione estremamente innovativa di IA che si sta diffondendo di recente è l'uso del reinforcement learning (RL) nella gestione del magazzino. Come discusso nel capitolo precedente, il RL è una particolare tecnica di machine learning in cui un agente apprende e adatta il proprio comportamento sulla base dei feedback ricevuti dall'ambiente in cui opera, a seconda dell'azione compiuta [60]. Le applicazioni di questa tecnologia alla logistica interna di magazzino sono numerose, dall'ottimizzazione delle rotte di picking al controllo dei sistemi di sorting, fino all'attività di packing. Il principale vantaggio derivante dall'uso del reinforcement learning è la sua capacità di tale algoritmo di adattarsi dinamicamente ai cambiamenti di contesto in tempo reale [61].

In parallelo, i digital twins stanno guadagnando terreno come una tecnologia rivoluzionaria per la gestione operativa del magazzino. I digital twins sono rappresentazioni virtuali di asset fisici e sistemi, che consentono la simulazione, il monitoraggio e l'ottimizzazione delle real-world operations [62]. Questa tecnologia offre ai manager la possibilità di simulare vari scenari, valutandone l'impatto sulle operazioni, con notevoli vantaggi nel

processo decisionale. La capacità di considerare molteplici opzioni prima di implementare una tecnologia fisica consente alle aziende di ridurre rischi e costi, testando virtualmente nuove soluzioni [63]. Un ulteriore beneficio dei digital twins è la possibilità di prevedere le prestazioni del magazzino e ottimizzare l'uso delle risorse in scenari mutevoli.

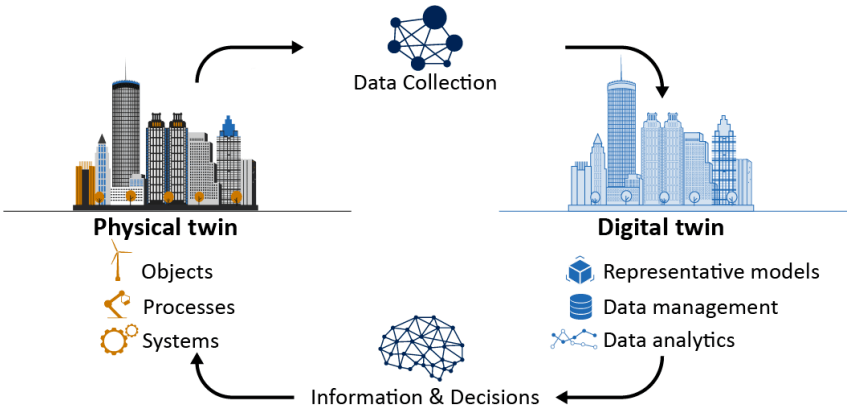

Figura 2.6: Esempio di digital twin.

L'implementazione di tecnologie basate su intelligenza artificiale nella gestione del magazzino offre notevoli benefici, ma introduce anche sfide che devono essere affrontate per integrarle con successo nelle operazioni quotidiane del centro di stoccaggio. Queste sfide includono considerazioni etiche, preoccupazioni sulla privacy, rischi di sicurezza e problematiche relative alla collaborazione uomo-macchina [64].
L'utilizzo di stistemi basati su IA solleva interrogativi riguardo all'etica e all'uso dei dati personali raccolti da questi algoritmi. Infatti, questi strumenti analizzano un volume considerevole di dati sensibili, di conseguenza garantirne la privacy diventa una priorità. I managersono quindi chiamati a implementare misure estremamente avanzate di cybersecurity, oltre a conformarsi a normative come il *General Data Protection Regulation (GDPR)*. Inoltre, un rischio significativo deriva dall'uso di dataset incompleti o non rappresentativi per l'addestramento, che potrebbero portare a decisioni discriminatorie [65]. Un esempio emblematico è l'uso di IA, daparte molte aziende, a supporto del processo di assunzione o di valutazione delle performance dei dipendenti.
La sicurezza è un ulteriore aspetto cruciale da considerare, in particolare durante la progettazione di un sistema di picking che richiede l'uso di robot per il prelievo. È essenziale prevenire collisioni, danni ai beni o incidenti che coinvolgono gli operatori umani. I manager, dunque, devono sviluppare dei protocolli di sicurezza basati su standard come la ISO 13849 e la ANSI/RIA R15.06, che regolano la sicurezza delle operazioni robotizzate nei magazzini.
Infine, un'ulteriore sfida riguarda l'elaborazione di interfacce user-friendlye e intuitive per facilitare l'interazione degli operatori con i sistemi automatizzati. Questo richiede

anche l'istituzione di corsi di formazione specifici per aiutare il personale a collaborare efficacemente con i robot e sfruttare al meglio le tecnologie avanzate implementate.

Capitolo 3

Smart waving: presentazione del progetto e analisi delle possibili applicazioni

Il picking rappresenta uno dei processi con impatto maggiore sull'operatività del magazzino, sia in termini di costi operativi, che di tempistiche di evasione degli ordini. Di conseguenza, è un tema ampiamente discusso e analizzato. Gli studi si sono focalizzati nella ricerca di una soluzione di routing ottimale, introducendo l'intelligenza artificiale nella definizione del percorso e applicando euristiche sempre più avanzate, e sulla collaborazione tra operatori umani e robot, con l'obiettivo di ridurre gli errori, garantendo maggiore accuratezza e precisione e riducendo le tempistiche necessarie.
Tuttavia, un aspetto meno esplorato dalla letteratura riguarda il livello a monte della definizione delle routing policies: la pianificazione dei task di picking (*planning task*). Un tipico processo di prelievo, infatti, si articola in più fasi:

1. Ricezione degli ordini.
2. Gli ottimizzatori non sono in grado di elaborare un numero troppo elevato di ordini, di conseguenza, si lavora tramite **waves**, per cui i sistemi selezionano un certo numero di picking order e li *pianificano* (*planning task*).
3. L'ottimizzatore procede a creare dei task di picking rispettando i vincoli fornitogli (es. volume, peso, numero di ordini) e minimizzando la distanza percorsa dall'operatore (*picking task*).
4. La lista di picking elaborata viene suddivisa tra gli operatori o tra i robot, che procedono all'attività di prelievo.

La gestione delle waves, sebbene sia fondamentale per distribuire il carico di lavoro e migliorare la fluidità delle operazioni, è un'attività che viene spesso trascurata e realizzata in modo poco strutturato, selezionando, ad esempio, gli ordini in modo arbitrario. Il planning, di conseguenza, risulta inefficiente, non tenendo conto di eventuali correlazione tra i picking order, e con buoni margini di miglioramento.

Il progetto dello sviluppo dello **Smart waving**, portato avanti con il supporto del team Data science di Logistic Reply s.r.l., si propone di affrontare queste inefficienze, con l'obiettivo di ottimizzare la generazione dei planning task. Tale strumento mira a migliorare la fase di pianificazione del picking, creando cluster di ordini, selezionati a partire da un set di picking order iniziale, ottimizzati non solo in base ai vincoli di peso, volume, data di spedizione (*due date*) e numero di ordini gestibili, ma anche per garantire maggiore uniformità all'interno dei raggruppamenti stessi. Questa uniformità facilita l'elaborazione successiva da parte dell'ottimizzatore, che potrà creare task di picking più efficienti e, in ultima analisi, ridurre i tempi di prelievo.
Il termine "Smart" implica una scelta di pianificazione non casuale, bensì un processo di selezionato più mirato e consapevole, in modo da sfruttare al meglio le risorse disponibili e di ridurre gli sprechi di tempo e denaro. Per questa ragione lo Smart waving si discosta dai metodi tradizionali, ricercando una composizione ottima di ciascuna wave, integrando informazioni aggiuntive e riducendo le disparità tra gli ordini assegnati a ogni cluster.

Il progetto è stato sviluppato all'interno della funzione ricerca e sviluppo dell'azienda, che si occupa di eslorare le opportunità dell'intelligenza artificiale nell'ambito della gestione del magazzino. Il team valuta l'applicazione dei più recenti trend di IA ai processi interi, tra cui l'uso dell'IA generativa a supporto delle decisioni, l'applicazione di algoritmi di machine learning al forecasting e lo studio dei possibili utilizzi del reinforcement learning. L'adozione del RL è proprio al centro di questa analisi, essendo una metodologia di recente sviluppo nell'ambito del ML, con un grande potenziale di applicazione ai processi di warehouse management. L'obiettivo è valutare i limiti e i benefici che caratterizzano questa famiglia di algoritmi, e le sue prospettive di integrazione con altri processi. Infatti, un aspetto estremamente interessante di questa metodologia è la possibilità di incorporarla ai processi di inbound e outobound logistics. L'impiego del reinforcement learning, del resto, non si limita all'ottimizzazione del picking, può estendersi anche ad altre aree critiche, come lo *yard management* - lo scheduling degli arrivi presso le varie baie di carico/scarico - la *cartonization*, ossia l'assemblaggio degli ordini in modo da massimizzare il riempimento dei contenitori e ridurre lo spreco di materiale. Inoltre, può essere utilizzato per risolvere lo *storage assignment problem*, migliorando l'efficienza del posizionamento degli articoli nei magazzini.
Il RL consente di ottenere performance migliori in gran parte degli ambiti strategici nella gestione del magazzino, portando numerosi benefici, che includono la capacità di

adattarsi dinamicamente ai cambiamenti dell'ambiente in cui opera, rendendo possibile un'ottimizzazione continua. Tuttavia, la sua implementazione presenta sfide significative. La progettazione di un modello di RL richiede dati di alta qualità, risorse computazionali avanzate e una notevole esperienza nel campo. Inoltre, un rischio importante è rappresentato dall'*overfitting*, ovvero la tendenza del modello ad adattarsi troppo ai dati di addestramento, riducendo la sua capacità di generalizzare e di operare in contesti nuovi.

Lo Smart waving rappresenta, quindi, una prima fase di studio di questi algoritmi nella gestione del magazzino. L'obiettivo è valutarne il valore aggiunto e le criticità, oltre a identificare i miglioramenti nell'efficienza del processo di picking. Questo potrebbe generare benefici significativi per le performance complessive del magazzino, rendendo la gestione della supply chain più agile e reattiva.

3.1 Reply: overview sul gruppo e focus su Logistics Reply

Logistics Reply è un delle companies appartenenti al gruppo Reply S.p.A., un network di aziende italiane altamente specializzate, che si distinguono nel campo della consulenza informatica e delle applicazioni di *digital services*. Il gruppo, quotato presso la Borsa Valori di Milano, opera in numerosi settori, tra cui: Telecom, Media & Entertainment (TME), logistica, finance & insuranse, settore pubblico, sanità e cybersecurity. La sua capacità di offrire soluzioni innovative in questi ambiti la rende un partner di riferimento per aziende che cercano di ottimizzare i loro processi e integrare tecnologie all'avanguardia.
Il gruppo Reply, a sua volta, è inglobata in Alika, una holding che detiene oltre il 50% dell'azienda. Tatiana e Filippo Rizzante, figli del fondatore Mario Rizzante, sono proprietari di circa il 13% delle quote di Alika, mantenendo un ruolo importante nella governance del gruppo.

Reply offre una vasta gamma di servizi, che spaziano dall'introduzione di intelligenza artificiale nei processi delle aziende clienti, alla cybersecurity, fino al gaming e molto altro. L'azienda si distingue per la sua natura estremamente innovativa, coniugando competenze di consulenza strategica e operativa, con avanzate capacità di sviluppo tecnologico. Infatti, investe notevoli risorse nella ricerca e nello sviluppo di nuove soluzioni tecnologiche, progettando e commercializzando prodotti propri. Alcuni esempi includono soluzioni avanzate di cybersecurity per la protezione di dati sensibili, l'implementazione di digital twins, l'automazione dei processi industriali, contribuendo alla trasformazione digitale dei propri clienti, senza dimenticare l'apporto nel campo dell'intelligenza artificiale, con soluzioni innovative di machine learning e big data analytics a supporto delle decisioni

aziendali. Questi elementi evidenziano come Reply si posizioni come un partner tecnologico capace di adattarsi alle sfide emergenti e di offrire soluzioni innovative ai propri clienti [66].

Figura 3.1: Logo di Reply S.pA.

Le iniziative legate alle tecnologie più all'avanguardia sviluppate dall'azienda vengono presentate e sperimentate all'interno dei loro *labs*, localizzati in alcune delle principali sedi del gruppo. Tra i più innovativi si distingue l'*Area42* di Torino, che ospita i progetti di più grande rilevanza dell'azienda, tra cui l'*autonomous warehouse lab*, dove vengono sviluppate e testate soluzioni autonome per la gestione delle operazioni di magazzino, mediante droni, computer vision esistemi WMS evoluti; il *robotics lab*, un ambiente dove viene valutata la capacità di robot autonomi nello svolger attività di patrolling e ispezione; infine, l'*industrial IoT lab*, un'area destinata alla progettazione di soluzioni innovative per l'automazione e la sostenibilità dei processi industriali. Questi ambienti riflettono la capacità di Reply di essere all'avanguardia nel settore tecnologico, mantenendo un costante aggiornamento sulle nuove tecnologie e anticipando le esigenze del mercato.

3.1.1 Accenni di storia

Reply nasce nel 1996 a Torino su iniziativa di Mario Rizzante, con una struttura originale: un network di società altamente specializzate, ciascuna operante in specifici ambiti, come il cloud computing, l'Internet of Things (IoT), e i digital media. Inizialmente, la presenza di Reply era limitata al mercato italian. Tuttavia, nel 2006, con Tatiana Rizzante come amministratore delegato, inizia un percorso di espansione in Europa, con particolare attenzione a mercati come il Regno Unito e la Germania. Filippo Rizzante, fratello di Tatiana, assume il ruolo di chief technology officer (CTO) del gruppo, contribuendo all'innovazione tecnologica dell'azienda.
Nel 2000, anno di quotazione in Borsa, la società aveva un fatturato che si attestava a 33 milioni di euro circa. La crescita, da allora, è costante, raggiungendo nel 2022, un fatturato di circa 2 miliardi di euro. Già nel 2004, *Forbes* aveva riconosciuto il potenziale della società, inserendola tra le 25 aziende italiane a maggiore tasso di crescita. Il gruppo Reply è ora uno dei leader del settore, diffondendosi in 18 paesi e con più di 15 mila persone. La nuova sede dell'azienda è localizzata a Milano.

Figura 3.2: Sede di Reply S.pA.

3.1.2 Logistics Reply: panoramica aziendale e piattaforma commercializzata

Logistics Reply è una delle aziende appartenenti al network del gruppo Reply S.p.A., specializzata nella consulenza in ambito logistico, con un focus particolare sulla gestione del magazzino. La sua missione è supportare le aziende nella trasformazione digitale della loro supply chain, offrendo soluzioni software innovative. L'azienda, fondata nel 1996, ha uffici nel Regno Unito, in Germania, negli Stati Uniti e in Brasile, con più di 200 clienti in tutto il mondo, tra cui nomi di rilievo come *GLS*, *Puma*, *Stellantis* e molti altri. L'obiettivo della società è quello di offrire soluzioni per garantire piena visibilità e tracciabilità sui processi logistici per l'azienda cliente.

Uno studio condotto da Gartner colloca Logistics Reply come una dei primi 18 fornitori globali di WMS, inserendola all'interno del *WMS Magic quadrant*, riconoscendola come un'azienda visionaria, prossima a emergere tra i leader del settore [67].

Figura 3.3: Magic quadrant di Gartner per i warehouse management system [67].

L'impegno di Logistics Reply nell'innovazione si manifesta anche attraverso investimenti significativi in ricerca e sviluppo, collaborazioni con università e istituti di ricerca per sviluppare soluzioni che anticipano i trend del settore. L'azienda è, inoltre, molto attenta alle nuove tecnologie, come l'intelligenza artificiale, il machine learning, e l'Internet of Things (IoT), integrandole nelle proprie soluzioni per migliorare l'efficienza e la sostenibilità delle operazioni logistiche.

L'attenzione alla sostenibilità è un altro pilastro della strategia di Logistics Reply. L'azienda si impegna a ottimizzare le operazioni logistiche per ridurre l'impatto ambientale, ad esempio, migliorando l'efficienza del trasporto e promuovendo l'uso di tecnologie a basso consumo energetico. Questo approccio non solo contribuisce a ridurre le emissioni di CO_2, ma si allinea anche con le crescenti aspettative dei clienti e del mercato verso una logistica più sostenibile.

LEA Reply™: la piattaforma per la gestione della supply chain

L'applicativo di più recente sviluppo è **LEA Reply™**, una suite lanciata nel 2016 inizialmente come piattaforma per il warehouse management, è stata poi estesa con l'aggiunta di nuovi moduli per rispondere alle mutevoli esigenze della supply chain. La piattaforma è progettata come un ecosistema modulare: i vari servizi possono essere combinati tra loro per creare soluzioni personalizzate che soddisfano le specifiche necessità dei clien-

ti. Ogni componente copre un'area critica della catena logistica, come la gestione dei fornitori tramite il supplier portal module, il coordinamento della logistica dell'ultimo miglio attraverso il last mile module, o l'ottimizzazione della preparazione degli ordini di e-commerce con l'in-store picking module.
L'uso di LEA Reply™ permette ai clienti di ottimizzare il flusso di lavoro, ridurre i tempi di inattività e migliorare la visibilità sui processi logistici, contribuendo così a decisioni più rapide e informate. La piattaforma è progettata per integrarsi facilmente con i sistemi IT esistenti, rendendo il processo di implementazione fluido e meno oneroso per le aziende.
L'uso di soluzioni avanzate come queste richiede elevati standard di sicurezza per proteggere i dati e le informazioni aziendali da possibili attacchi o fughe di informazioni. Per questo motivo, Logistics Reply adotta misure di sicurezza all'avanguardia, garantendo anche una rapida ripresa in caso di guasti hardware o software, assicurando la continuità operativa [68].

Figura 3.4: Servizi offerti interni alla piattaforma LEA Reply™.

3.2 LEA Reply™ Optimizer: funzionamento e limitazioni

Il problema del picking viene affrontato in varie modalità da Logistics Reply, avvalendosi di molteplici strumenti. Uno di questi è il **LEA Reply™ Optimizer**, software in grado di ottimizzare l'assemblaggio i task di prelievo mediante un approccio euristico. LEA Reply™ Optimizer è un tool basato su *Optaplanner*, un *constraint solver* basato su IA integrato nella piattaforma LEA Reply™.
L'Optaplanner consente la creazione di micro-servizi che operano all'interno della suite LEA Reply™ e possono essere utilizzati sia da applicazioni interne che esterne al sistema. Il servizio, accessibile tramite API REST, può essere configurato per risolvere problemi di ottimizzazione sulla base di parametri di input, adattabili a vari contesti operativi, regolando anche il tempo di esecuzione.

Una delle implementazioni di LEA Optimizer è il modulo di *Cluster Picking Optimization Service*, che mira a ottimizzare il processo di cluster picking, minimizzando la distanza totale percorsa dagli operatori. Il modulo prende in considerazione vari parametri di input, tra cui la disposizione delle ubicazioni in magazzino, le specifiche dei carrelli/contenitori, le strategie di allocazione, l'ottimizzazione dei percorsi di picking e altre caratteristiche

del processo. Il tool, avendo a disposizione questi dati, è in grado di riorganizzare i job di picking in un numero ottimale di task, riducendo al minimo la distanza complessiva percorsa.
Un esempio dell'uso di questo strumento è illustrato nella figura 3.5, dove i cluster di ordini sono visualizzati con diversi colori. Si può osservare come, senza l'utilizzo dell'Optimizer, i raggruppamenti sono in numero maggiore e non tengono conto di alcuni vincoli di trasporto che potrebbero consentire una movimentazione più efficiente. Al contrario, l'Optimizer tiene conto di questi fattori, riducendo il numero di gruppi di picking a due, semplificando così il processo di prelievo e migliorando la gestione operativa complessiva.

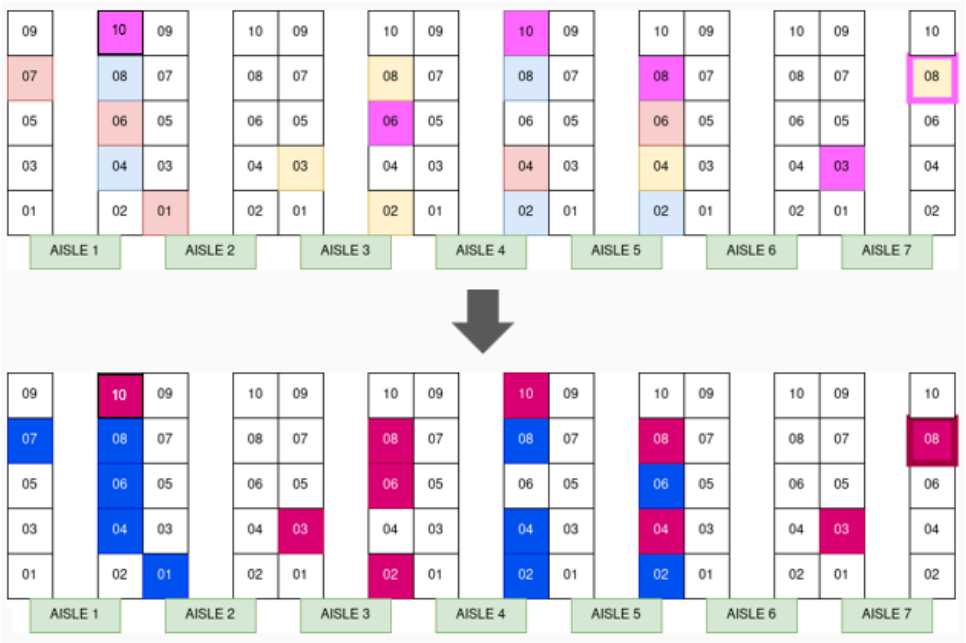

Figura 3.5: Esempio di situazione con e senza l'uso dell'Optimzier.

L'implementazione di questo servizio all'interno del processo di picking rende la gestione di questa attività più semplice ed efficiente, modificando parzialmente le modalità operative. La piattaforma di WMS, priva dell'Optimizer, gestisce i task di picking nel seguente modo:

1. Import degli ordini a flusso continuo.
2. Selezione degli ordini da pianificare.
3. **Stock allocation strategy**: selezione casuale dello stock, con logiche specifiche di assegnazione a determinate zone.
4. Creazione dei job di preliveo.
5. Creazione dei picking task con logica di aggregazione.

L'introduzione dell'Optimizer aggiunge uno step ulteriore successivo alla creazione dei picking job: una volta generati, viene inviata una richiesta al tool, il quale individua la combinazione ottimale di questi ultimi, restituendo i task di prelievo ottimizzati.

I benefici derivanti dall'uso di questo servizio sono molteplici: si riduce il numero di corridoi visitati in media da ciascun operatore in un pick-tour, si ottimizzano i raggruppamenti per una maggiore efficienza in termini di numero di articoli prelevati in un singolo pick-tour e si minimizza la distanza complessiva percorsa dai picker.

Questi aspetti positivi sono evidenti in un caso studio condotto da Logistics Reply per un'azienda cliente operante nel settore del fashion. L'analisi è stata fatta su uno dei magazzini del cliente, confrontando la situazione con e senza l'Optimizer. Nel secondo scenario mancava una logica di raggruppamento dei job basata sulla vicinanza delle ubicazioni, con un impatto negativo sull'efficienza del prelievo. Nel corso dello studio, è stato analizzato il centro di stoccaggio, osservando l'importanza del concetto di "modulo", ossia gruppo di corridoi, e l'adozione della S-shape routing policy.
Le metriche considerate nell'analisi si suddividono in due categorie principali: *count*, che stima il numero di aree univoche visitate dai picker, e *range*, che misura la vastità dell'area di magazzino coperta durante il picking. Questi indicatori possono essere calcolati in termini di gruppi di corridoi univoci visitati all'interno dello stesso task (*aisle group count/range*) oppure in termini di coppie modulo-corridoio univoche visitate (*aisle count/range*). Un'ulteriore metrica rilevante è la *location count*, che tiene conto delle singole ubicazioni.

Group Count	Group Range	Aisle Count	Aisle Range	Location Count
17.07%	16.16%	17.21%	0.44%	0.01%
10.53%	12.52%	17.34%	2.31%	0.27%
13.04%	20.00%	17.71%	12.28%	2.41%
11.91%	14.37%	19.29%	3.17%	0.19%
20.83%	31.25%	21.28%	23.87%	3.45%
11.88%	18.20%	19.98%	0.61%	0.13%
16.00%	23.53%	15.07%	13.16%	1.22%
12.46%	18.28%	20.72%	5.37%	0.51%
0.00%	0.00%	0.00%	0.00%	0.00%
10.28%	17.22%	18.61%	3.65%	-0.95%

Figura 3.6: Risultati dello studio effettuato.

I risultati, come illustrato nella figura 3.6, evidenziano un miglioramento in ciascuna delle metriche prese in analisi nelle diverse waves testate durante lo studio.

Nonostante i numerosi benefici derivanti dall'implementazione dell'Optimizer all'interno del proprio processo di picking, questo strumento presenta alcune limitazioni. In particolare, la creazione dei cluster di ordini si basa su un'euristica che, per sua natura, non garantisce l'individuazione della soluzione ottimale. Di conseguenza, vi è margine per ulteriori miglioramenti, ad esempio tramite l'utilizzo di metodologie più avanzate e innovative. Un altro aspetto da considerare è il tempo di esecuzione del servizio, che può superare i 10 minuti, risultando non trascurabile per alcune operazioni. Tuttavia, la principale criticità di tale strumento risiede nella gestione delle waves: come discusso in precedenza, la selezione degli ordini viene fatta in modo casuale, spesso considerando i primi ordini disponibili a sistema, a causa di una visibilità limitata sugli ordini. Questo approccio può generare inefficienze che lo Smart waving mira a eliminare, migliorando la pianificazione e la gestione delle attività di prelievo.

3.3 Valutazione dei vari casi d'uso presi in considerazione

Lo Smart waving è una delle possibili applicazioni del reinforcement learning al processo di prelievo, ottimizzando il planning dei task di picking. In fase di analisi sono stati presi in considerazione due ulteriori casi d'uso per l'implementazione del RL nella gestione del magazzino: l'ottimizzazione dei percorsi di picking e la risoluzione del storage assignment problem. Questi scenari di utilizzo evidenziano la flessibilità di questa tecnica, in quanto l'impiego di tale famiglia di algoritmi permetterebbe di raggiungere prestazioni migliori rispetto a quelle che si potrebbero ottenere con i metodi tradizionali.

3.3.1 Primo caso d'uso: ottimizzazione dei picking path

L'ottimizzazione dei path di picking è uno dei temi maggiormente approfonditi in letteratura riguardante la logistica di magazzino. La ricerca si è focalizzata sull'identificazione di nuove tecniche o metodologie per ridurre al minimo la distanza percorsa dagli operatori in fase di prelievo. Il reinforcement learning è un approccio estremamente innovativo e di recente sviluppo che si adatta a contesti dinamici e in continuo cambiamento. La definizione dei percorsi di picking è una delle applicazioni maggiormente analizzate dagli studiosi per questa famiglia di algoritmi. Grazie alla sua capacità di adattamento, l'RL è stato ampiamente studiato per la definizione dei percorsi di picking, offrendo prestazioni spesso superiori rispetto ai metodi tradizionali.
Uno studio in particolare ha esplorato l'integrazione del reinforcement learning nell'ottimizzazione dei percorsi di prelievo in un magazzino, la cui configurazione è mostrata in figura 3.7 [69].

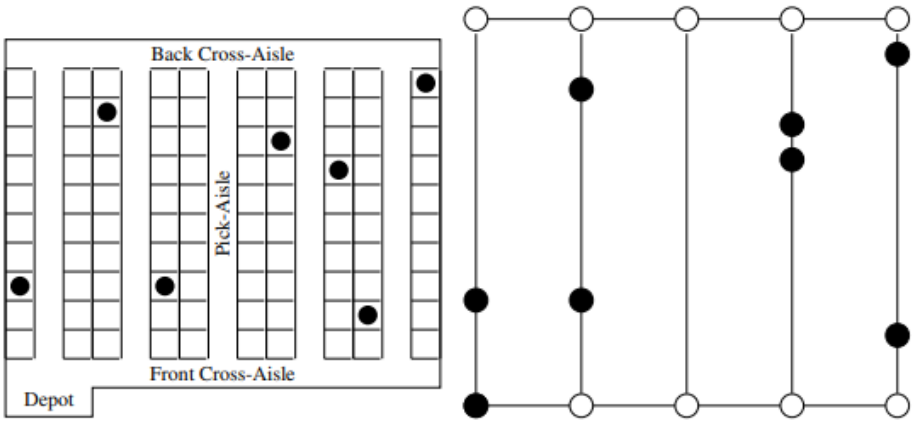

Figure 1: Stardard Rectangular Warehouse

Figura 3.7: Pianta del magazzino analizzato nello studio [69].

Il problema è stato formulato come un *travelling salesman problem (TSP)*, dove a ogni iterazione l'algoritmo seleziona uno dei nodi non ancora visitati, rappresentando le diverse ubicazioni dei prodotti. La selezione di un nodo ha un costo, determinato dalla distanza tra esso e la posizione corrente dell'agente. Una peculiarità dello studio risiede nell'adozione della distanza di Manhattan al posto di quella euclidea, per adattarsi meglio alla configurazione del magazzino.
La valutazione delle performance del RL è stata fatta tramite un confronto con le euristiche tradizionali di routing: S-shape, return, largest gap e composite, descritte nel capitolo precedente. I risultati, riportati nei grafici in figura 3.8, mostrano la distanza dalla soluzione ottimale per ciascun metodo all'aumentare del numero di corridoi. L'algoritmo di RL si distingue per la sua capacità di avvicinarsi maggiormente alla soluzione ottima rispetto agli approcci tradizionali, dimostrando il suo potenziale per ottimizzare le operazioni di picking.

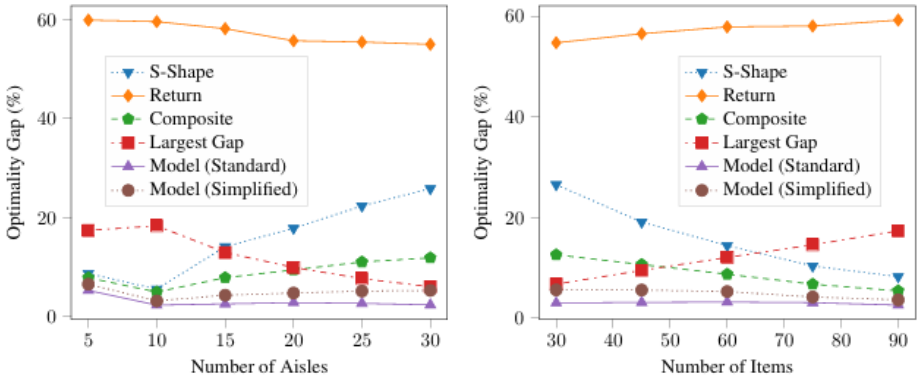

Figura 3.8: Distanza media dall'ottimo del modello proposto, confrontato con le euristiche più comuni per le varie classi di problemi, ossia in base ai numero di corridoi (sinistra) e alla dimensione della lista di prelievo (destra) [69].

Un ulteriore studio propone la modellizzazione del processo di picking tramite un problema di *two-sided online bipartite matching (TOBM)*. Il TOBM è una variante del tradizionale problema di matching bipartito, in cui un set di entità noto deve essere assegnato dinamicamente a un differente insieme non conosciuto a priori, i cui oggetti vengono mostrati sequenzialmente, uno per volta, con l'obiettio di massimizzare determinate metriche, che possono essere diverse a seconda del contesto. La complicazione del two-sided implica che entrambi i gruppi di nodi possano giungere in modo sequenziale, incrementando la complessità. L'analisi applica questo concetto alla collaborazione tra operatori umani e AMR nel processo di picking.
Lo studio effettua un confronto tra le euristiche tradizionali e il modello di reinforcement learning, considerando le medesime strategie di routing. La valutazione è stata fatta utilizzando un environment di dimensioni ridotte, dove eseguire il training, quattro ambienti di dimensione sempre maggiore dove svolgere i test e un ultimo per analizzare il tempo di computazione [70].

Environment name	**Grid size**	**|P|**	**N_o**	**Training**	**Testing**
Env 1	6×6	2	1	True	True
Env 2	10×10	6	3	False	True
Env 3	10×10	6	1	False	True
Env 4	20×20	24	12	False	True
Env 5	40×40	96	48	False	True
Env 6[4]	80×80	384	192	False	True

Figura 3.9: Configurazione degli ambienti considerati. P è il numero di operatori e N_0 il massimo numero di nuovi ordini per istante di tempo [70].

Alcune definizioni utili a comprendere al meglio l'analisi svolta:

- Ciascun AMR viene denominato "ordine" e può trovarsi in tre stati: announced, ongoing or tardy. Nel momento in cui l'ordine entra nello stato ongoing potrà essere raccolto.
- C_H il costo legato a ciascun periodo di tempo in cui un ordine ongoing non viene collezionato. Se non viene raccolto entro la sua due date, il suo stato diventa tardy, a cui viene associato un costo $C_T > C_H$.

La valutazione, dunque, viene fatta in termini di minimizzazione della funzione di costo.

I risultati, anche in questo caso, mostrano come il reinforcement learning sia decisamente superiore alle euristiche, sia nell'ambiente di training sia in quelli di test, come si evince anche dalla figure sottostanti.

Nell'ambiente 6, infine, è stata effettuata una valutazione riguardo il tempo di calcolo di ciascun algoritmo. I risultati mostrano come all'aumentare delle dimensioni del problema,

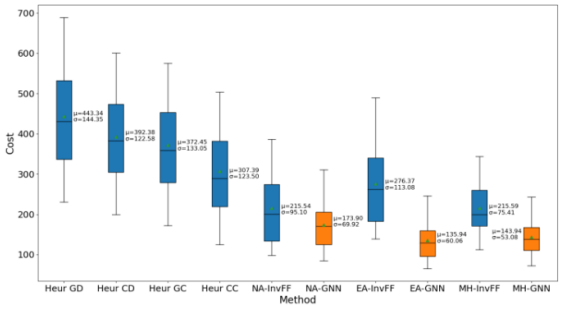

Figura 3.10: Prestazioni delle varie metodologie nell'ambiente di test [70].

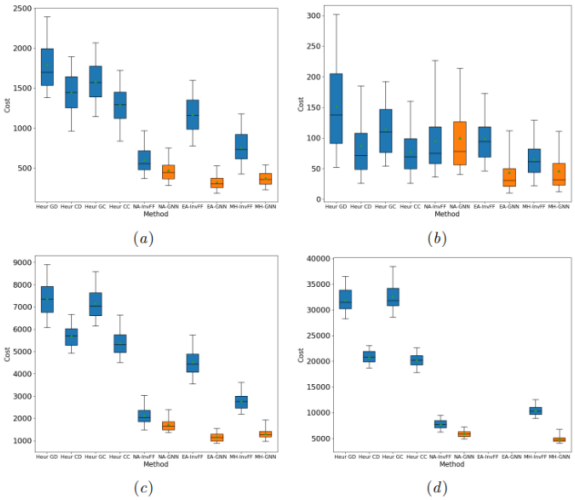

Figura 3.11: Prestazioni delle varie metodologie nei successivi quattro ambienti [70].

cresce anche il tempo necessario per calcolare le azioni per ogni metodo. Mentre le euristiche sono molto più veloci degli approcci DRL su piccole configurazioni, questo divario si riduce costantemente, fino al punto in cui la scelta diventa indifferente, di seguito una rappresentazione grafica di tali risultati.

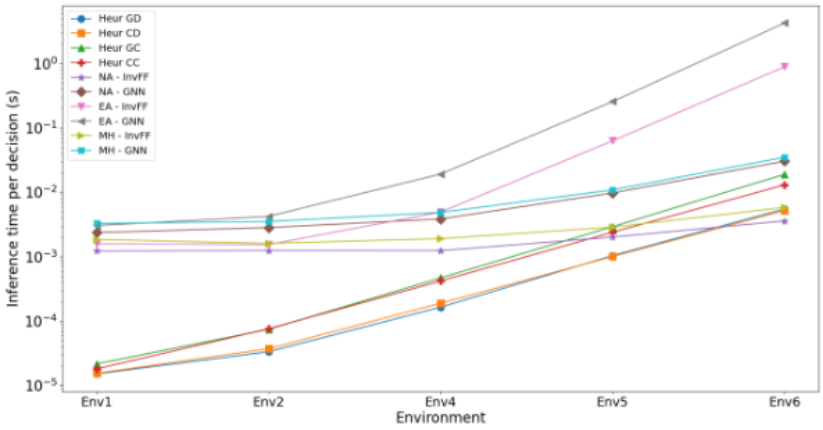

Figura 3.12: Tempo computazionale richiesto per ciascun metodo [70].

L'applicazione del reinforcement learning alla fase di routing rappresenta una valida soluzione, con notevoli benefici: vi è ampia letteratura disponibile in merito, l'implementazione non si ritiene essere particolarmente complessa, comporta un notevole miglioramento nelle prestazioni dell'attività di picking e viene identificato come un'area efficace per introdurre il RL all'interno della gestione del magazzino e in seguito estenderla ad ambiti ad esso collegati. Tuttavia, La limitazione principale è rappresentata dall'esistenza del Cluster Optimizer, che porterebbe a un *overlapping* di servizi. Di conseguenza, si è deciso di optare per una soluzione differente.

3.3.2 Secondo caso d'uso: storage assignment problem

Lo storage assignment problem (SAP) è un tema strettamente connesso all'ottimizzazione delle operazioni di picking, poichè riguarda l'assegnazione ottimale delle ubicazioni degli articoli in magazzino. Una corretta attribuzione degli spazi permette una riduzione della distanza percorsa dagli operatori, migliorando l'efficienza delle operazioni di prelievo e riducendo i tempi complessivi di esecuzione. In letteratura, il SAP è stato ampiamente discusso, con varie soluzioni che suggeriscono l'implementazione di tecnologie avanzate per migliorare le performance. Il reinforcement learning trova naturale applicazione in questo ambito, grazie alla sua capacità di adattarsi a scenari dinamici. Numerosi studi evidenziano i vantaggi di questa metodologia applicata al SAP.
Uno studio in particolare utilizza una rete neurale per prevedere gli ordini futuri e impiega un algoritmo di RL per bilanciare l'efficienza nelle tempistiche di picking con i costi operativi, in un'implementazione semplificata del SAP [71]. La tabella 3.1 riassume le variabili considerate nello studio:

Class	**Variable**	**Notation**	**Range**
State	Order probability	p_{it}	[0, 1]
	Order	o_{it}	$\{0, 1\}$
	Storage map	map_t	Permutation([M])
Action	Reposition	a_{it}	$\{(k, l) \mid 1 \leq k < l \leq M\} \cup \{\text{None}\}$
Reward	Total travel time	$c_{travel,t}$	R_+
	Reposition cost	$c_{reposition,t}$	R_+

Tabella 3.1: Descrizione delle variabili e del loro range di variabilità [71].

Il modello proposto dagli autori si basa su due assunzioni fondamentali:

1. Il costo di refilling del prodotto è nullo.
2. La routing policy è fissa ed è riassunta dalla funzione *travel()*

Un elemento chiave del modello è rappresentato dal sistema di forecast della domanda, che opera da filtro per i cambiamenti di stato e riduce la varianza della reward. Le prestazioni dell'algoritmo sono state testate in due ambienti distinti: uno statico (*setting 1*) e uno dinamico (*setting 2*), come riportato nella tabella 3.2.

Setting 1

Il modello è stato confrontato con la soluzione ottimale tabulare, considerando un numero ridotto di 10 prodotti. Questa scelta è stata dettata dalla limitata scalabilità della soluzione tabulare, che risulta impraticabile su un set di dati più ampio. La curva di apprendimento dell'algoritmo proposto, rappresentata in figura 3.13 (linea blu), evidenzia

Environment Variables	Setting 1	Setting 2
Number of Products	10	10
Layout	2×5	2×5
γ	0.99	0.99
Order Process	Static	Semiannual
True Order Probability	[0.1, 0.19, 0.28, 0.37, 0.46, 0.54, 0.63, 0.72, 0.81, 0.9]	Season 1: [0.1, 0.19, 0.28, 0.37, 0.46, 0.54, 0.63, 0.72, 0.81, 0.9] Season 2: [0.9, 0.81, 0.72, 0.63, 0.54, 0.46, 0.37, 0.28, 0.19, 0.1]
Season length	-	500 periods

Tabella 3.2: Setting degli ambienti nei due scenari [71].

come l'algoritmo raggiunga rapidamente elevate reward, convergendo verso la soluzione ottimale tabulare.

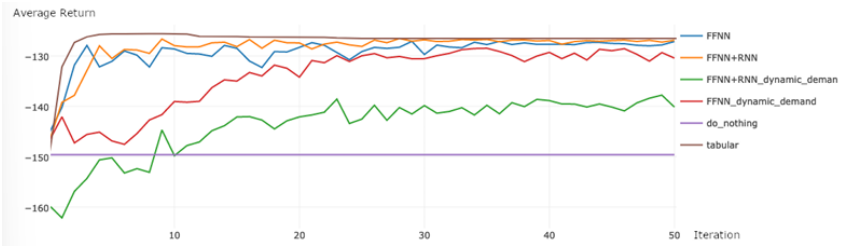

Figura 3.13: Curva di apprendimento di vari modelli e della soluzione tabulare ottima [71].

I risultati ottenuti nella valutazione del trade-off tra tempi di percorrenza e costi operativi sono mostrati in figura 3.14. Sebbene il tempo di percorrenza risulti leggermente superiore rispetto alla soluzione ottimale, l'algoritmo proposto riesce a contenere i costi operativi, risultando dunque una soluzione efficiente dal punto di vista economico.

(a) Total Reposition Costs

(b) Total Travel-time Costs

Figura 3.14: Breakdown dei costi [71].

Setting 2

In un contesto dinamico, la probabilità di ordine varia seguendo un pattern specifico, come indicato in tabella 3.2. La curva di apprendimento dell'algoritmo in questo contesto è rappresentata in figura 3.13 (linea verde e linea rossa). Sebbene l'algoritmo non raggiunga

la soluzione ottimale, il risultato ottenuto è comprensibile data la natura dinamica del problema: l'algoritmo non dispone di informazioni complete riguardo alle variazioni reali delle probabilità di ordine. Analogamente, nella valutazione del trade-off tra tempo di percorrenza e costi operativi, la linea verde di figura 3.14 evidenzia le prestazioni dell'algoritmo, mostrando una soluzione che, pur essendo subottimale, risulta efficace in un contesto dove le probabilità cambiano continuamente.

L'uso del reinforcement learning nello storage assignment problem offre notevoli benefici, come l'assegnazione quasi ottimale delle ubicazioni agli articoli, il che si traduce in una significativa riduzione dei tempi di relievo, senza un eccessivo aumento dei costi operativi. Tuttavia, esistono diverse limitazioni importanti da considerare per questo tipo di applicazione. La complessità di implementazione è elevata e richiede un periodo di formazione per gli operatori, affinché possano adattarsi all'uso di questi sistemi avanzati. Inoltre, esiste il rischio che l'applicazione del RL in questo contesto rimanga solo teorica: l'implementazione pratica potrebbe richiedere una completa riorganizzazione del layout di magazzino dell'azienda. Questi fattori hanno portato alla decisione di considerare questo tipo di applicazione solo in una fase successiva, concentrandosi, invece, sullo Smart waving come priorità immediata.

Capitolo 4

Smart waving: progettazione del modello per la pianificazione dei task di picking

Lo Smart waving rappresenta una soluzione innovativa per affrontare il problema della pianificazione dei task di picking, selezionando gli ordini ottimali sulla base di parametri specifici forniti in input. Il principale vantaggio di questo modello risiede nella sua flessibilità e scalabilità, consentendo di adattarsi alle esigenze aziendali più disparate. I parametri, che includono dati relativi a volumi, pesi, date di spedizione, e priorità degli ordini, possono variare a seconda delle necessità delle aziende clienti, garantendo l'adattabilità di tale algoritmo a diversi contesti aziendali.
Un altro aspetto chiave dello Smart waving è la sua dinamicità. Grazie all'utilizzo di algoritmi di reinforcement learning, il modello è in grado di migliorare costantemente le proprie performance adattandosi ai cambiamenti nelle condizioni operative, rendendo la gestione del magazzino più efficiente. Questo lo rende particolarmente adatto a contesti in cui la variabilità operativa e la complessità logistica richiedono soluzioni dinamiche e veloci.
Lo Smart waving, inoltre, supera le criticità del metodo attualmente in uso, che spesso presenta limiti nel numero massimo di ordini gestibili in una singola ondata (wave). L'algoritmo sviluppato non è soggetto a questo vincolo e consente una pianificazione più fluida e ottimizzata, riducendo i colli di bottiglia e migliorando il flusso complessivo delle operazioni di picking. Tuttavia, esistono alcune limitazioni all'implementazione di questo algoritmo, rendendo complessa una sua introduzione all'interno dei processi di magazzino. La progettazione dell'algoritmo, in particolare, rappresenta una delle principali sfide nell'implementazione di questa metodologia, necessitando di un'analisi approfondita dei suoi elementi chiave. La funzione di reward deve essere definita in modo da garantire un

apprendimento efficace da parte dell'agente. L'individuazione degli stati dell'ambiente, necessari al modello per compiere delle scelte che mirino a un continuo miglioramento, deve essere fatta in modo che rifletta con precisione la complessità operativa del magazzino. Lo spazio delle azioni possibili, infine, deve essere formulato in modo che l'agente possa modificare l'ambiente ricercando un miglioramento complessivo. Inoltre, il tempo di calcolo è un fattore da non trascurare, con l'aumentare del numero di ordini da gestire, il tempo richiesto per il training e per la convergenza a una soluzione valida può crescere esponenzialmente, influendo sull'efficienza del sistema.

L'obiettivo finale dello Smart waving è individuare l'istante ottimale per procedere alla pianificazione dei picking order, con lo scopo di massimizzare l'efficienza operativa del magazzino. Il modello non solo determina il momento ideale per la pianificazione, ma si occupa anche della creazione di cluster di ordini, la cui esecuzione sarà ottimizzata tramite l'Optimizer. Questo approccio permette di ridurre drasticamente la frequenza di esecuzione delle waves, garantendo un flusso continuo di task di picking. In tal modo, si minimizzano i tempi di inattività e si evita di sovraccaricare gli operatori, migliorando significativamente la produttività complessiva. Il processo si articola, dunque, come segue:

1. Flusso di ordini in arrivo.
2. Ricerca del raggruppamento ottimale e pianificazione del task.
3. Processo iterativo: : raggiunta la soglia di ordini (es. 10 ordini), l'algoritmo valuta se sono adatti a realizzare un cluster ottimale o meno, in caso positivo viene realizzato e inviato all'optimizer per la ricerca del percorso ottimo, in caso contrario l'algoritmo rimane in attesa di ordini aggiuntivi che gli permettano di realizzare un raggruppamento ottimale.

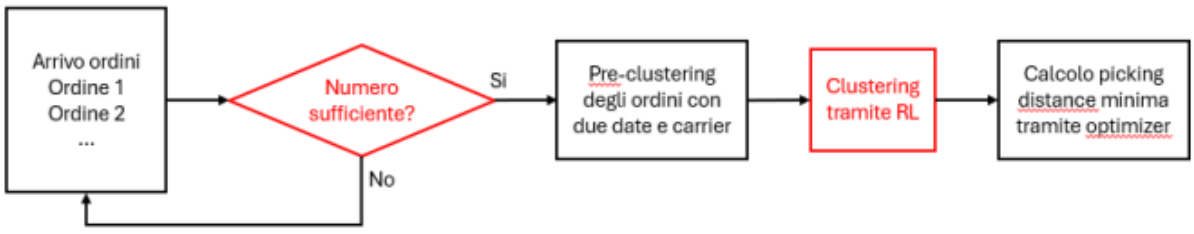

Figura 4.1: Flow chart descrittivo del processo di Smart waving.

Le sezioni successive tratteranno nel dettaglio l'analisi di fattibilità del progetto, la fase di design dell'algoritmo, esplorando gli elementi costitutivi del modello, e le valutazioni fatte relative ai livelli successivi del processo di sviluppo, riguardanti la scelta dell'ambiente di simulazione.

4.1 Analisi preliminare di fattibilità

L'attuale processo di pianificazione del picking in azienda si articola attraverso due fasi principali, ognuna delle quali si avvale di strumenti distinti:

1. Pianificazione dei picking job tramite Automatic Planning: a partire dalla lista degli ordini, seguendo picking profiles, due date e carrier assegntai, viene pianificata l'assegnazione dei picking job.

2. Creazione dei picking task: questa fase può essere gestita tramite due approcci distinti: la *planning strategy* o l'Optimizer.

    - **Planning strategy**: è un tool integrato all'interno dell'automatic planning, che realizza un clustering degli ordini, assegnandoli a task di picking e associandoli a una *handling unit (HU)*, secondo parametri predefiniti. Tuttavia, l'ordine di prelievo è assegnato in modo casuale e non tiene conto di alcun percorso ottimale per la raccolta.

    - **Optimizer**: è un tool separato dal planning, che ha lo stesso ruolo della planning strategy, ma, in aggiunta, determina il percorso ottimale per il prelievo degli ordini, utilizzando algoritmi euristici.

L'introduzione del reinforcement learning all'interno del processo di planning ha richiesto un'approfondita analisi del flusso di pianificazione attuale, valutando nel dettaglio il suo funzionamento e gli elementi che lo compongono. La fase iniziale del processo riguarda l'analisi dei dati sugli ordini forniti in input e la successiva elaborazione dei *picking profiles*. Un picking profile rappresenta un insieme omogeneo di righe d'ordine, definito secondo attributi specifici dell'azienda cliente, come B2B o B2C, e configurato con un timing che stabilisce la frequenza di pianificazione. Questo sistema permette di organizzare le waves in funzione delle necessità operative del magazzino (ad esempio, ordini e-commerce al mattino e B2B al pomeriggio), ottimizzando la gestione delle operations. Una volta definiti i picking profiles, si procede all'applicazione delle *allocation rules*, per poi creare i task di picking finali.

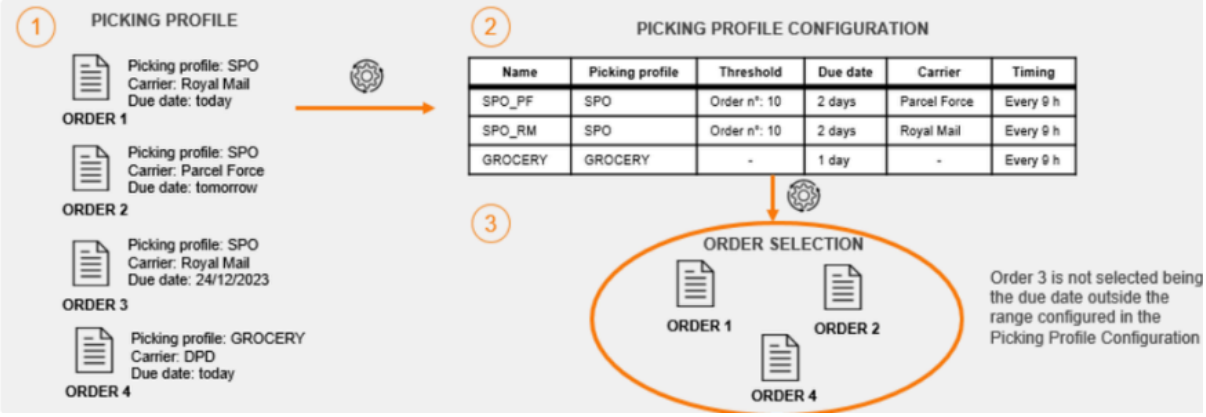

Figura 4.2: Esempio di generazione dei picking order a partire da picking profile, due date e carrier di ogni ordine.

L'obiettivo del progetto è, quindi, la realizzazione di un modello che sia in grado di raggruppare in modo ottimale gli ordini di picking da pianificare, in funzione di parametri prestabiliti per minimizzare il tempo di picking.
Attualmente, l'attenzione dell'analisi è focalizzata sul processo di clustering degli ordini, con lo sviluppo di uno strumento capace di ottimizzare i raggruppamenti, integrando un numero sempre crescente di parametri. A questo fine, è stato condotto uno studio approfondito dei diversi metodi di machine learning e loro applicazioni nel clustering di oggetti. La ricerca è stata condotta sul supervised learning, l'unsupervised learning e il reinforcement learning (RL), valutando come l'intelligenza artificiale possa essere introdotta in questo contesto per migliorare le prestazioni del sistema.

4.1.1 Implementazione di un modello di unsupervised learning

I metodi di unsupervised learning, come discusso nei capitoli precedenti, vengono utilizzati per analizzare grandi set di dati alla ricerca di pattern, con il vantaggio di non richiedere il supporto di un operatore umano. La naturale applicazione di queste tecniche è il clustering di dati statici. Esiste un gran numero di algoritmi di clustering basati sull'unsupervised learning, tra cui i più diffusi sono il *K-means* e l'*agglomerative hierarchical clustering (AHC)*.
Il K-means algorithm si articola in diversi passaggi: : inizialmente si stabilisce il numero di cluster desiderati, assegnando i Kcentroidi iniziali. Successivamente, ciascun punto del dataset viene associato al centroide più vicino, utilizzando come criterio la distanza, spesso la distanza euclidea. Una volta assegnati tutti i punti, i centroidi vengono ricalcolati come la media dei punti appartenenti a ciascun cluster. Questo processo di riassegnazione e ricalcolo dei centroidi continua iterativamente finché non si ottiene una stabilizzazione delle posizioni dei centroidi o si raggiunge un numero massimo di iterazioni.
L'AHC, d'altra parte, opera seguendo una metodologia differente: ogni punto del set di

dati viene considerato come un singolo cluster e si procede al calcolo delle distanze tra tutte le coppie di cluster. I due raggruppamenti più vicini vengono uniti e le distanze vengono ricalcolate. Il processo si ripete fino a che tutti i punti non sono stati uniti in un unico grande cluster. La definizione della distanza tra i cluster può variare, a seconda del criterio adottato.

Queste metodologie sono comunemente utilizzate in numerosi settori, grazie alla loro efficacia nel rilevamento di pattern nei dati. Tuttavia, presentano una limitazione singificativa: sono più adatti ad operare in contesti statici, che non evolvono nel tempo. Qualora questa condizione venisse meno sarebbe necessario procedere al riaddestramento del modello sul nuovo set di dati, ogni qualvolta questo viene modificato, portando a un consumo di tempo significativo. Inoltre, un ulteriore criticità è rappresentata dalla necessità di definire un numero di cluster all'inizio del processo. Ciò non è possibile nel caso analizzato. Per questi motivi, queste soluzioni non risultano ideali per il problema in esame, che richiede un approccio dinamico e in grado di adattarsi continuamente alle variazioni operative.

4.1.2 Implementazione di un modello di supervised learning

Il supervised learning è una tecnica di machine learning che apprende dai dati etichettati forniti in input, identificando pattern all'interno del dataset. La differenza con la metodologia vista in precedenza, è la necessità dell'intervento di un operatore umano, fornendo all'algoritmo coppie input-output attesi e monitorando le prestazioni del modello durante la fase di training. Esistono vari studi in letteratura che trattano l'ottimizzazione del clustering mediante tecniche di supervised learning. Ad esempio, uno studio rilevante discute l'uso di *algoritmi genetici (GA) con feedback iterativo* applicati a una popolazione di soluzioni, la quale viene migliorata attraverso operazione genetiche come selezione, crossover e mutazione. Questo processo iterativo permette di esplorare ampi spazi di soluzioni e convergere verso l'ottimo. Nel caso in esame, i GA possono adattarsi dinamicamente ai cambiamenti nei dati e alle condizioni operative, migliorando progressivamente le soluzioni di clustering. I GA, inoltre, valutano le soluzioni sulla base di una funzione di fitness, che nel contesto del problema potrebbe essere correlata al tempo di picking. Inoltre, i GA valutano le soluzioni basate su una funzione di fitness, che nel problema in esame potrebbe essere comparato al tempo di picking [72].
Un'altra metodologia rilevante è rappresentata dal *simulated annealing (SA) con obiettivi multipli*. Un'applicazione specifica di questa tecnica al clustering mostra come, tramite un processo iterativo, sia possibile migliorare la soluzione riducendo gradualmente una temperatura simulata. A ogni iterazione, una nuova soluzione vicina viene valutata e accettata o rifiutata in base alla differenza di costo e alla temperatura corrente. Analogamente al problema dell'ottimizzazione del picking, il SA è utilizzato per risolvere problemi

complessi, dove la funzione obiettivo potrebbe includere sia il tempo di prelievo che altri parametri operativi [73].

L'implementazione di un modello di supervised learning nel clustering degli ordini rappresenta una valida soluzione. Si tratta di tecniche innovative, che conducono a risultati soddisfacenti in contesti statici. Tuttavia, non sono esenti da problematiche: sebbene siano efficaci in contesti statici, con dataset che non evolvono nel tempo, in un ambiente dinamico come quello del magazzino potrebbero non performare al meglio. Inoltre, la fase di training di tali modelli richiede grandi voluie di dati storici, spesso non facilmente reperiribili, oltre a essere un processo dispendioso in termini di tempo. Per queste ragioni, l'attenzione si è spostata verso metodologie più flessibili, capaci di adattarsi meglio alle variazioni operative.

4.1.3 Implementazione di un modello di reinforcement learning

Il reinforcement learning è una tecnica di machine learning estremamente innovativa, basata su un processo di apprendimento iterativo attraverso trial-and-error. L'agente interagisce con l'ambiente e riceve feedback, sotto forma di ricompense o penalizzazioni, che guidano la sua evoluzione verso soluzioni ottimali. Le sue applicazioni al clustering sono molteplici, in vari ambiti di ricerca. Numerosi studi in letteratura evidenziano l'efficacia del RL nell'affrontare problemi complessi legati al raggruppamento di entità. Ad esempio, uno studio propone l'uso del RL per risolvere il problema del *load balancing* nel cloud computing. In questo contesto, l'algoritmo si occupa di suddividere le virtual machine in due gruppi: overloaded e underloaded. L'utilizzo del RL permette una redistribuzione dinamica del carico tra queste entità, tenendo in considerazione metriche come il *makespan*, il consumo energetico e l'utilizzo della CPU. I risultati mostrano che l'integrazione di RL con algoritmi di ottimizzazione ibridi avanzati, come il *HLFO - Hybrid Lyrebird Falcoln Optimization*, porta a un significativo miglioramento nell'efficienza del clustering [74].
Un ulteriore studio rilevante analizza l'uso del reinforcement learning nell'ambito delle reti radio cognitive (CRN), introducendo degli schemi di routing basati sul clustering. L'indegine svolta mostra come il clustering e il RL risolvono il problema del routing nelle CRN, migliorando la scalabilità e la stabilità della rete. Il modello propostoè stato testato attraverso simulazioni, mostrando come il clustering basato su RL riduca al minimo le interferenze tra gli utenti secondari (SU) e quelli primari (PU), e selezioni percorsi di routing più stabili. Questi risultati dimostrano che il clustering basato su RL non solo migliora l'efficienza delle reti radio cognitive, ma consente anche una maggiore stabilità operativa rispetto ai metodi tradizionali [75].

I numerosi studi a disposizione dimostrano che il reinforcement learning è una soluzione

altamente versatile e efficace per il clustering, in grado di gestire dinamicamente parametri operativi come il tempo di picking, oltre a non richiedere dati storici per il training. Questi vantaggi rendono il RL una metodologia particolarmente adatta a contesti operativi dinamici, consentendo il raggiungimento di soluzioni vicine all'ottimale, un traguardo che altri metodi di machine learning difficilmente riescono a conseguire. Tuttavia, esistono alcune limitazioni all'implementazione di questo algoritmo, la principale risiede nella necessità di disporre di un ambiente di simulazione adeguato per il training dell'algoritmo. Inoltre, rispetto ai metodi di supervised e unsupervised learning, il RL è più complesso da sviluppare, richiede un periodo di training più lungo e un impegno maggiore nella fase di progettazione del modello.

4.1.4 Confronto tra i modelli proposti

I modelli proposti rappresentano soluzioni valide al problema del clustering degli ordini, ciascuno con benefici e criticità. La decisione è stata presa valutando approfonditamente diversi fattori, come riportato nella tabella 3.3.

	Reinforcement learning	**Supervised learning**	**Unsupervised learning**
Near-optimal solutions	SI	NO	NO
Implementation effort	ALTO	MEDIO	BASSO
Dynamic adaptability	SI	NO	NO
External parameters (picking time)	SI	NO	NO
Historical data required	NO	SI	NO
Simulation environment necessary	SI	NO	NO

Tabella 4.1: Confronto tra reinforcement learning, supervised learning e unsupervised learning.

Come si evince dalla tabella sopra riportata, il RL rappresenta la soluzione migliore per il caso in analisi, le principali limitazioni sono rappresentate dalla necessità di un ambiente di simulazione in cui effettuare il training dell'algoritmo e dall'elevato effort richiesto per una sua implementazione, oltre ad un tempo computazionale considerevole. Tuttavia, per affrontare quest'ultima sfida, è possibile adottare alcune soluzioni che mirano a migliorare l'efficienza senza compromettere la qualità del risultato:

- Pre-clustering con Filtri di Riduzione della Complessità: per ridurre il numero di ordini da considerare in ogni iterazione, in modo da diminuire il numero di iterazioni necessarie per ottenere un buon risultato. Ad esempio, si potrebbe procedere a un raggruppamento preliminari degli ordini basato su criteri semplici come la data di

spedizione o la classificazione ABC degli articoli, prima di applicare algoritmi di clustering più complessi.

- Algoritmi di Clustering Veloci: come K-Means++ o Mini-Batch K-Means, progettati per lavorare su grandi dataset, potrebbe fornire una soluzione efficace in tempi più brevi..
- Parallelizzazione e Distribuzione del Calcolo: sfruttare tecniche di parallelizzazione per eseguire il clustering e l'ottimizzazione del percorso su diversi segmenti di dati in parallelo.

In conclusione, data la presenza dell'Optimizer e la richiesta da parte del management della coesistenza con i tool già in uso, si ritiene che l'applicazione del RL alla pianificazione dei picking task possa essere la soluzione migliore come prima implementazione di un algoritmo di questo tipo all'interno del processo di outbound.

4.2 Componenti costitutivi del modello - Action space, observation space e reward function

La fase progettuale successiva ha incluso le attività legate al design del modello di reinforcement learning. Gli algoritmi di RL condividono una struttura di base comune, composta da alcuni elementi costitutivi: la *policy*, la *reward function*, l'*observation space* e l'*action space*.
La *policy* guida l'agente nelle sue decisioni, mappando gli stati alle azioni e determinando quale azione l'agente debba eseguire in ogni stato, con l'obiettivo di massimizzare la ricompensa accumulata nel tempo. Esistono diverse strategie di policy come le deterministiche (che associano una specifica azione a ciascuno stato) e quelle stocastiche (che assegnano una probabilità alle possibili azioni da eseguire in uno stato). La selezione della policy giusta dipende dalle caratteristiche del problema e verrà discussa più dettagliatamente nel capitolo successivo.
Il modello apprende tramite un processo di trial-and-error, che rende la progettazione della reward function, dell'observation space e dell'action space cruciali per il successo dell'algoritmo.

La funzione di reward segnala all'agente la bontà dell'azione compiuta, mostrando se ha migliorato o peggiorato la situazione corrente, rappresentano, così, un meccanismo di feedback continuo. L'obiettivo dell'agente è massimizzare questo valore sul lungo termine, seguendo una fase iniziale di esplorazione delle azioni possibili, per poi selezionare in modo ripetitivo quelle che, in passato, hanno generato le reward maggiori. La progettazione di tale funzione porta con sè alcune sfide particolarmente complesse. In primo luogo, è necessario individuare le variabili rilevanti per il processo di decisione, poiché queste

devono essere legate a obiettivi concreti e misurabili. Inoltre, è importante bilanciare il tipo di feedback: una funzione di ricompensa lineare può semplificare l'apprendimento, ma potrebbe non catturare la complessità di alcuni scenari operativi. Funzioni di ricompensa non-lineari offrono maggiori possibilità di personalizzazione e possono dare priorità a determinate variabili rispetto ad altre (ad esempio, priorizzare la riduzione del tempo di prelievo rispetto alla riduzione del consumo energetico). Infine, un'altra questione rilevante è la definizione dei pesi da attribuire a ciascuna variabile. La selezione di pesi appropriati dipende dal contesto e dagli obiettivi specifici dell'azienda, ad esempio, bilanciando tra velocità di esecuzione, accuratezza e costi operativi.

L'observation space rappresenta tutte le variabili utili a descrivere lo stato dell'ambiente in un determinato istante temporale, permettendo all'agente di acquisire le informazioni necessarie per compiere decisioni efficaci. Definire accuratamente questo spazio è fondamentale: dati incompleti o irrilevanti potrebbero compromettere il processo di apprendimento, portando a decisioni subottimali. Nel contesto del picking, potrebbe includere dati come la posizione degli articoli in magazzino, il numero di operatori disponibili, la capacità dei mezzi di movimentazione e altre informazioni logistiche. Una delle principali sfide legate alla definizione dell'observation space riguarda la qualità dei dati. I dati raccolti devono essere accurati, aggiornati e coerenti con la situazione reale. Errori o ritardi nell'acquisizione di questi dati potrebbero influenzare negativamente il comportamento dell'agente, che si troverebbe a prendere decisioni basate su informazioni obsolete o imprecise. Inoltre, occorre bilanciare la quantità di informazioni fornite all'agente: se l'observation space è troppo ridotto, l'agente non avrà abbastanza dati per prendere decisioni informate; se è troppo ampio, invece, il tempo di calcolo e la complessità computazionale aumenteranno notevolmente.

Infine, l'action space rappresenta le possibili azioni che l'agente può eseguire per modificare lo stato dell'ambiente, generando una reward positiva o negativa. Una corretta definizione delle azioni è essenziale per il processo di apprendimento del modello. Le azioni devono essere sufficientemente granulari per consentire all'agente di esplorare efficacemente lo spazio delle soluzioni, ma allo stesso tempo non devono essere troppo complesse da rendere il calcolo inefficiente. Ad esempio, in un contesto di picking, le azioni potrebbero includere la selezione del prossimo ordine da processare, la scelta del percorso da seguire, o la riorganizzazione dinamica degli ordini in base alle priorità logistiche. Individuare azioni che siano ripetibili in diversi contesti è una sfida non banale nella progettazione del modello. La scelta delle azioni deve tenere conto delle dinamiche del magazzino, della configurazione fisica, della disponibilità di risorse come mezzi di movimentazione e operatori, e dei vincoli temporali imposti dalle scadenze di consegna. Un'errata selezione delle azioni potrebbe portare a decisioni subottimali da parte dell'agente, rallentando il processo di apprendimento e compromettendo l'efficacia complessiva del modello. Inoltre,

l'action space può essere discreto o continuo, a seconda del tipo di problema che si vuole risolvere. Un action space discreto offre un numero limitato di azioni predefinite, mentre uno continuo permette una gamma infinita di scelte. La decisione su quale tipologia di action space utilizzare dipende dal contesto operativo specifico: ad esempio, se l'obiettivo è scegliere tra un numero finito di percorsi, un action space discreto potrebbe essere sufficiente, mentre per ottimizzazioni più complesse, come la gestione della velocità dei mezzi di trasporto, potrebbe essere preferibile un action space continuo.

4.2.1 Definizione della reward function del modello

La funzione di reward fornisce un feedback all'agente sulla qualità dell'azione compiuta e sul suo impatto sull'environment in cui opera. La definizione di tale funzione è uno dei passaggi più complessi nella progettazione di un modello di reinforcement learning, poiché richiede una profonda comprensione del contesto aziendale e dei fattori che influiscono sulle performance. Nel contesto del clustering, la reward function deve incentivare l'agente a prendere decisioni mirate all'ottimizzazione delle variabili considerate, in funzione dei pesi attribuiti a ciascuna. Nel modello proposto, le principali variabili che influenzano la definizione della reward includono:

- *peso totale degli ordini del cluster*;
- *volume complessivo degli articoli*;
- *numero di ordini presenti nel raggruppamento*;
- *deviazione standard della data di spedizione (due date)*;
- *deviazione standard dei corridoi (aisle)*;
- *deviazione standard delle campate (span)*;

L'obiettivo è stato quello di garantire dei valori vicini al massimo trasportabile dagli operatori per i primi tre parametri, in modo da ottenere cluster efficienti e gestibili in termini di carico, e di minimizzare i restanti tre, per ridurre la distanza percorsa dai picker e ottimizzare le operazioni di prelievo. Nel corso del progetto sono stati valutati due diversi approcci alla funzione di reward con risultati differenti: una funzione matematica unica oppure una composizione di funzioni rappresentanti i vari parametri del cluster.

Funzione unica aggregata

Questo approccio combina tutte le variabili in una singola funzione matematica, attribuendo a ciascuna un peso in base alla sua importanza all'interno del processo. Questa metodologia consente all'agente di avere un'unica metrica di riferimento, facilitando la valutazione delle prestazioni, ma rischia di penalizzare eccessivamente alcune variabili

se il bilanciamento dei pesi non è calibrato con precisione. La funzione risultante è la seguente:

$$R_i = b{\cdot}V_{dev,i}+c{\cdot}P_{dev,i}+d{\cdot}(1-\sigma_{D,norm,i})+e{\cdot}cSIZE_{dev,i}+f{\cdot}(1-\sigma_{S,norm,i})+g{\cdot}(1-a_{A,norm,i})$$

$$V_{dev,i} = e^{-\left(\frac{V_i - V_{MAX}}{V_{MAX}}\right)^3} \quad P_{dev,i} = e^{-\left(\frac{P_i - P_{MAX}}{P_{MAX}}\right)^3}$$

$$\sigma_{D,norm,i} = \frac{\sigma_{D,i} - \sigma_{D,min}}{\sigma_{D,MAX} - \sigma_{D,min}} \quad \sigma_{S,norm,i} = \frac{\sigma_{S,i} - \sigma_{S,min}}{\sigma_{S,MAX} - \sigma_{S,min}}$$

$$cSIZE_{dev,i} = e^{-\left(\frac{cSIZE_i - cSIZE_{MAX}}{cSIZE_{MAX}}\right)^2} \quad \sigma_{A,norm,i} = \frac{\sigma_{A,i} - \sigma_{A,min}}{\sigma_{A,MAX} - \sigma_{A,min}}$$

$$N_{dev,i} = e^{-\left(\frac{N_i - N_{MAX}}{N_{MAX}}\right)^2}$$

dove:

- *a,b,c,d,e,f* sono i pesi che vengono attribuiti a ciascun componente della funzione (utilizziamo gli stessi per tutti i cluster). Possono variare tra 1 e 5.
- V_i è il volume degli ordini raggruppati nell'i-esimo cluster.
- V_{MAX} è il massimo volume degli ordini raggruppabili in un cluster.
- P_i è il peso degli ordini raggruppati nell'i-esimo cluster.
- P_{MAX} è il massimo peso degli ordini raggruppabili in un cluster.
- $cSIZE_i$ è il numero degli ordini raggruppati nell'i-esimo cluster.
- $cSIZE_{MAX}$ è il massimo numero degli ordini raggruppabili in un cluster.
- $\sigma_{D,i}$ è la deviazione standard della due date nell'i-esimo cluster.
- $\sigma_{D,min}$ è la minima deviazione standard della due date tra tutti i cluster.
- $\sigma_{D,MAX}$ è la massima deviazione standard della due date tra tutti i cluster.
- $\sigma_{S,i}$ è la deviazione standard dello span nell'i-esimo cluster.
- $\sigma_{S,min}$ è la minima deviazione standard dello span tra tutti i cluster.
- $\sigma_{S,MAX}$ è la massima deviazione standard dello span tra tutti i cluster.

- $\sigma_{A,i}$ è la deviazione standard della spanPosition nell'i-esimo cluster.
- $\sigma_{A,min}$ è la minima deviazione standard della spanPosition tra tutti i cluster.
- $\sigma_{A,MAX}$ è la massima deviazione standard della spanPosition tra tutti i cluster.
- N è il numero di cluster effettuati.
- N_{MAX} è il numero di cluster realizzabili. Tale valore può essere definito a seconda del numero di operatori disponibili e della loro produttività.

La funzione di reward così definita incentiva l'agente a minimizzare le metriche di deviazione standard relative alla data di spedizione, ai corridoi e alle campate, normalizzando i risultati. Per quanto riguarda gli indicatori di volume, peso e dimensione del cluster, l'agente mira a raggiungere di un valore pari o vicino al massimo trasportabile da un operatore. La reward function è composta da sotto-funzioni non lineari, per consentire una migliore prioritizzazione dei parametri. Inoltre, il valore ottenuto viene, poi, normalizzato per rientrare in un intervallo facilmente gestibile dal modello.

Questa struttura della funzione di reward è stata ritenuta adatta a un'implementazione successiva, avviando una prima fase di test con una versione semplificata, al fine di ridurre la complessità per validare il modello.

Composizione di funzioni

Questo secondo approcco considera la composizione di funzioni per ottenere una reward function valida, separando i parametri in base alle diverse componenti. Tale modalità rappresenta una semplificazione della funzione unica, agevolando così il training del modello. La funzione è espressa come segue:

$$R = a_{norm} \cdot R_{DD} + b_{norm} \cdot R_A + c_{norm} \cdot R_S + d_{norm} \cdot R_V + e_{norm} \cdot R_P + f_{norm} \cdot R_{cSIZE}$$

dove R_{DD} è definito come:

$$R_{DD} = 1 - \frac{\text{DEVSTD}(DD)}{\text{DEVSTD}(DD)_{\max}}$$

con:

$$a = 8, \quad a_{norm} = 0.235$$

BONUS: se DEVSTD$(DD) \leq 4$, allora $\sqrt{a_{norm}}$

Per la componente R_A:

$$R_A = 1 - \frac{\text{DEVSTD}(A)}{\text{DEVSTD}(A)_{\max}}$$
$$b = 3, \quad b_{norm} = 0.088$$

BONUS: se DEVSTD$(A) \leq X$, allora $\sqrt{b_{norm}}$

Per la componente R_S:

$$R_S = 1 - \frac{\text{DEVSTD}(S)}{\text{DEVSTD}(S)_{\max}}$$
$$c = 3, \quad c_{norm} = 0.088$$

BONUS: se DEVSTD$(S) \leq X$, allora $\sqrt{c_{norm}}$

Per la componente R_V:

$$R_V = \begin{cases} -1 & \text{se } V > V_{max} \\ 1 \text{ e } \sqrt{d_{norm}} & \text{se } V_{max}(1-k) \leq V \leq V_{max} \\ \frac{V}{V_{max}} & \text{se } V < V_{max} \end{cases}$$
$$d = 6, \quad d_{norm} = 0.176$$

Per la componente R_P:

$$R_P = \begin{cases} -1 & \text{se } P > P_{max} \\ 1 \text{ e } \sqrt{e_{norm}} & \text{se } P_{max}(1-k) \leq P \leq P_{max} \\ \frac{P}{P_{max}} & \text{se } P < P_{max} \end{cases}$$
$$e = 6, \quad e_{norm} = 0.176$$

Per la componente R_{cSIZE}:

$$R_{cSIZE} = \begin{cases} -1 & \text{se } cSIZE > cSIZE_{max} \\ 1 & \text{se } cSIZE_{max}(1-k) \leq cSIZE \leq cSIZE_{max} \\ \frac{cSIZE}{cSIZE_{max}} \text{ e } f^2_{norm} & \text{se } cSIZE < cSIZE_{max} \end{cases}$$
$$f = 8, \quad f_{norm} = 0.235$$

dove k rappresenta un valore di tolleranza pari al 10%.

La reward function così definita è composta da sottofunzioni non complesse, con valori compresi tra -1 e 1, facilitando l'agente nella gestione dei risultati ottenuti. Ciascuna componene è associata a un *bonus*, assegnato all'agente quando raggiunge prestazioni ottimali. Questa modalità di operare incentiva maggiormente il modello a seguire la direzione desiderata. Inoltre, è stata valutata la possibilità di normalizzare i pesi, affinchè la reward complessiva non sia mai superiore a uno. Tuttavia, questa funzione non è priva di limitazioni: la sua applicazione è limitata a scenari semplici e alla fase iniziale del training. Qualora si desideri raggiungere prestazioni superiori, risulterà conveniente adottare funzioni simili a quella precedentemente descritta.

4.2.2 Definizione dell'action space del modello

Lo spazio delle azioni raccoglie tutte le possibili azioni che l'agente può compiere per influenzare l'environment. La complessità nella definizione di questo aspetto del modello riguarda l'individuazione di azioni che siano ripetibili in contesti diversi e che offrano un grado sufficiente di granularità per permettere un'adeguata esplorazione da parte dell'agente. Le azioni inizialmente individuate sono state ritenute troppo specifiche e limitanti, non permettendo all'agente di ricercare nuove soluzionie orientandolo eccessivamente verso l'obiettivo. Di seguito vengono elencate le azioni inizialmente considerate:

- Selezionare uno degli ordini con dueDate maggiore di 1.5 volte la media delle dueDate del cluster A e spostarlo in un cluster in cui la dueDate dell'ordine selezionato è compresa tra 1.5 volte la media delle dueDate del cluster e 0.75 volte la media delle dueDate del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.
- Selezionare uno degli ordini con dueDate minore di 0.75 volte la media delle dueDate del cluster A e spostarlo in un cluster in cui la dueDate dell'ordine selezionato è compresa tra 1.5 volte la media delle dueDate del cluster e 0.75 volte la media delle dueDate del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.
- Selezionare uno degli ordini con span maggiore di 1.5 volte la media degli span del cluster A e spostarlo in un cluster in cui lo span è compreso tra 1.5 volte la media degli span del cluster e 0.75 volte la media degli span del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.
- Selezionare uno degli ordini con span minore di 0.75 volte la media degli span del cluster A e spostarlo in un cluster in cui lo span è compreso tra 1.5 volte la media degli span del cluster e 0.75 volte la media degli span del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.

- Selezionare uno degli ordini con aisle maggiore di 1.5 volte la media degli aisle del cluster A e spostarlo in un cluster in cui il valore dell'aisle dell'ordine scelto è compreso tra 1.5 volte la media degli aisle del cluster e 0.75 volte la media degli aisle del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.
- Selezionare uno degli ordini con aisle minore di 0.75 volte la media degli aisle del cluster A e spostarlo in un cluster in cui il valore dell'aisle dell'ordine scelto è compreso tra 1.5 volte la media degli aisle del cluster e 0.75 volte la media degli aisle del cluster, altrimenti se non esiste nessun cluster in cui tale condizione è verificata, crea un nuovo cluster.
- Spostare un ordine da uno tra i cluster con un numero di ordini superiore al massimo accettabile ad uno tra i cluster con il numero di ordini maggiore rispetto agli altri, ma sempre inferiore al massimo accettabile, altrimenti se tutti hanno un numero di ordini superiori al massimo, crea un nuovo cluster.
- Spostare un ordine da uno tra i cluster con un volume superiore al massimo accettabile ad uno tra cluster con volume maggiore rispetto a tutti gli altri, ma sempre inferiore del massimo accettabile, altrimenti se tutti hanno volume superiore al massimo, crea un nuovo cluster.
- Spostare un ordine da uno tra i cluster con un peso superiore al massimo accettabile a uno tra i cluster con peso maggiore rispetto a tutti gli altri, ma sempre inferiore del massimo, altrimenti se tutti hanno peso superiore al massimo crea un nuovo cluster.

Questo approccio può sembrare valido in una prima fase di sviluppo, ma si è rivelato troppo restrittivo per l'agente. Pertanto, è stato deciso di abbandonarlo quasi subito, spostandosi verso una modalità che consente un ampio margine decisionale al modello:

- Selezionare un ordine dalla lista.
- Spostare l'ordine all'interno dell'i-esimo cluster.
- Creare un nuovo cluster.
- Scambiare due ordini tra due cluster diversi.

Questa modalità si è dimostrata più efficace per un corretto apprendimento da parte dell'agente, offrendo azioni ripetibili in contesti diversi. Un ulteriore passo in avanti è stato compiuto a seguito del cambiamento nell'approccio al problema, che verrà descritto nel capitolo successivo. Le azioni definitive realizzabili dal modello sono quindi le seguenti:

- Selezionare un ordine dalla lista.

- Spostare l'ordine all'interno del cluster.

4.2.3 Definizione dell'observation space del modello

L'observation space raccoglie tutte le informazioni utili al modello per prendere decisioni accurate. Una definizione corretta di tale spazio è essenziale per garantire un buon processo di apprendimento dell'agente. La complessità risiede nell'individuare quante e quali variabili siano necessarie a supportare il processo decisionale. Nel caso in analisi, le variabili considerate riguardano i dati relativi agli ordini e ai cluster creati:

- V_i
- P_i
- $cSIZE_i$
- $DEVSTD(DD)_i$
- $DEVSTD(S)_i$
- $DEVSTD(A)_i$
- N
- Codice identificativo degli ordini presenti all'interno dei cluster.

A seguito dell'implementazione del nuovo approccio al problema, le osservazioni relative al numero di cluster creati e ai codici identificativi degli ordini presenti nei cluster non sono state ritenute più rilevanti ai fini del progetto.

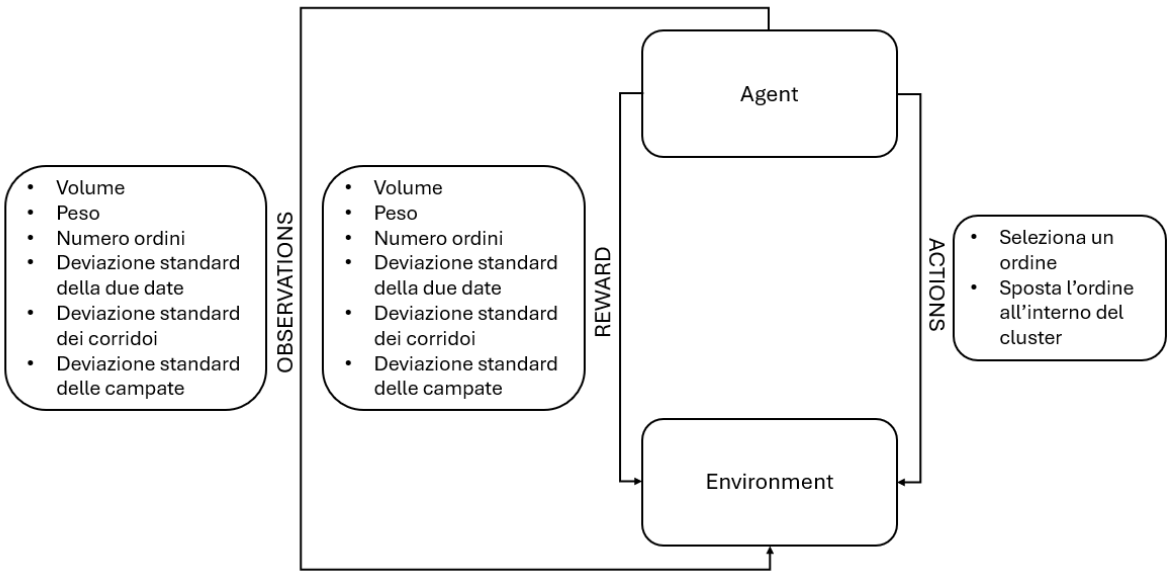

Figura 4.3: Rappresentazione del funzionamento del modello con variabili considerate.

4.3 Valutazione dei vari ambienti di simulazione

L'implementazione di un algoritmo di RL richiede, tipicamente, lo sviluppo di un ambiente di simulazione, che rappresenta l'environment in cui il modello opera.Questi ambienti sono essenziali per accelerare il processo di apprendimento dell'algoritmo, permettendo al modello di testare e migliorare le proprie strategie in un contesto sicuro e controllato, senza rischi associati a test nel mondo reale. Inoltre, un ambiente di simulazione offre flessibilità nel definire condizioni diverse e scenari complessi, il che è fondamentale per permettere all'agente di esplorare soluzioni ottimali. Tuttavia, creare tali ambienti richiede risorse aggiuntive e competenze specialistica, differenti da quelle richieste per la progettazione del modello di reinforcement learning. Nel corso del progetto, sono stati valutati tre tra i principali applicativi disponibili: *Gymnasium OpenAI*, *AWS RoboMaker* e *NVIDIA Omniverse Isaac Sim*, ciascuno con i suoi punti di forza e limitazioni specifiche.

Gymnasium OpenAI è un tool open-source molto diffuso, che fornisce una piattaforma per lo sviluppo e il test di algoritmi RL. Sebbene non offra ambienti predefiniti per applicazioni di magazzino, Gymnasium permette una facile creazione di nuovi ambienti personalizzati. Attualmente, viene ampiamente utilizzato per lo sviluppo di videogiochi. La sua semplicità e flessibilità lo rendono particolarmente adatto per le prime fasi di sviluppo e per scenari di test più semplici. Tuttavia, man mano che il modello si evolve e richiede simulazioni più complesse, Gym potrebbe risultare limitato e necessitare di un passaggio a soluzioni più avanzate [76].

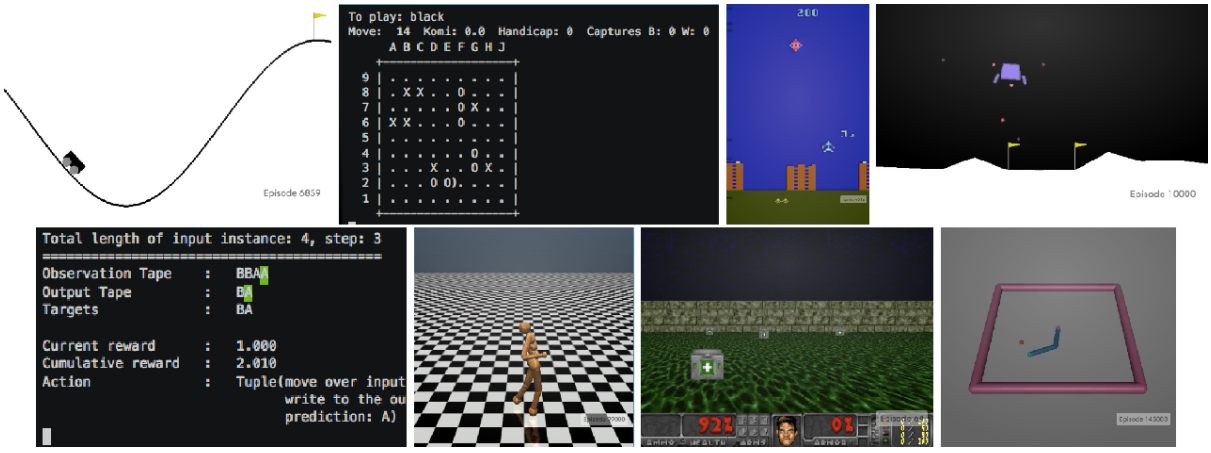

Figura 4.4: Esempi di ambienti sviluppati con Gymnasium OpenAI [76].

Amazon Robotic Warehouse Simulator (AWS RoboMaker) è un servizio di simulazione basato sul cloud, realizzato da Amazon, pensata specificatamente per lo sviluppo, il test e il deployment di applicazioni robotiche. Si integra con ROS (Robot Operating System), il sistema operativo per robot, e offre una piattaforma scalabile per eseguire simulazioni in parallelo. Questo strumento si distingue per la sua scalabilità e per la

possibilità di ridurre il carico hardware locale, grazie alla sua esecuzione su infrastrutture cloud. RoboMaker semplifica il processo di configurazione e offre servizi di gestione per l'integrazione continua e il deployment (CI/CD), il che lo rende un'opzione ideale per progetti che richiedono simulazioni su larga scala e con molte variabili in gioco. Tuttavia, potrebbe essere più complesso da configurare rispetto a soluzioni più leggere come Gym, rendendolo più adatto a fasi avanzate del progetto [77].

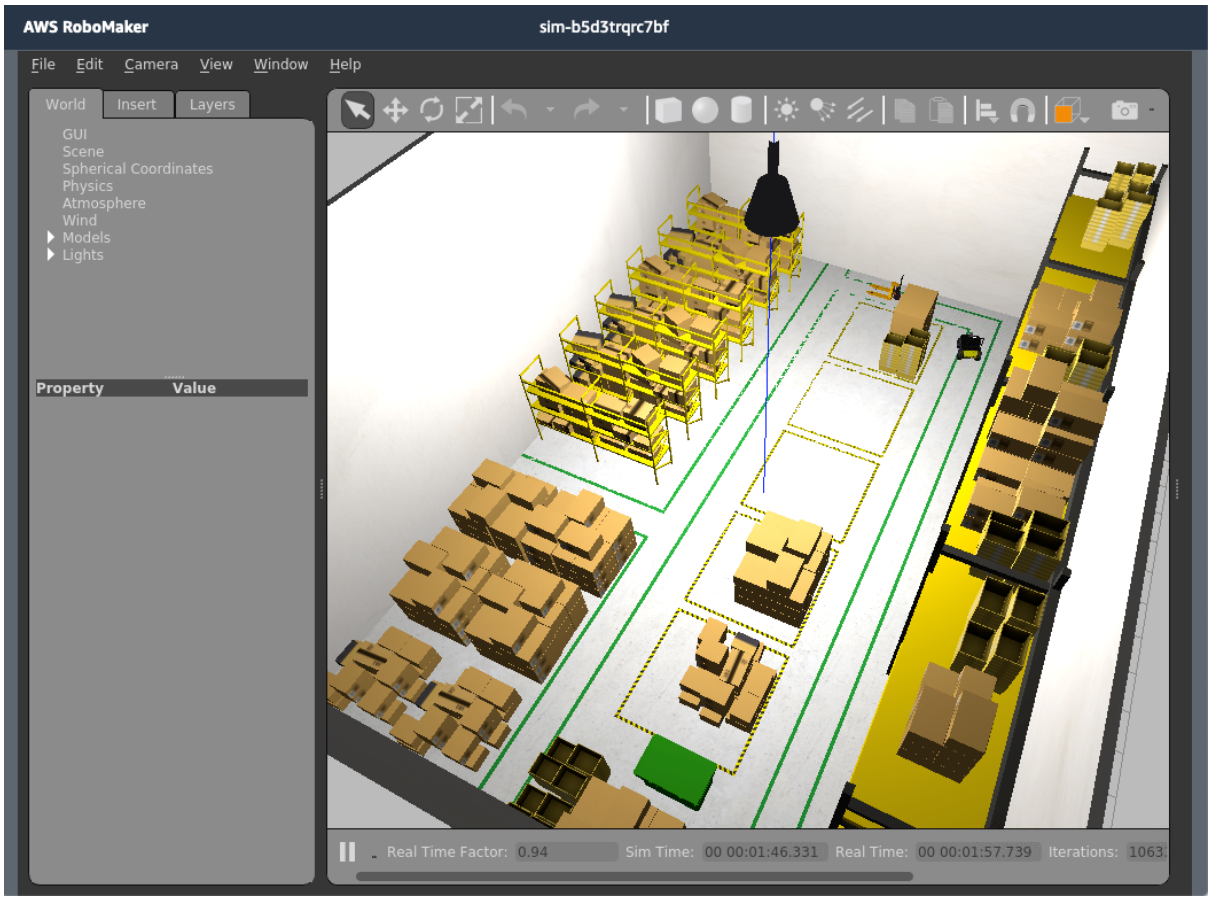

Figura 4.5: Esempio di ambiente di magazzino sviluppato con AWS RoboMaker [77].

NVIDIA Omniverse Isaac Sim rappresenta una soluzione estremamente avanzata per l'addestramento tramite reinforcement learning in contesti di robotica. Utilizzando il motore fisico PhysX di NVIDIA, consente simulazioni fisiche rapide e dettagliate, mantenendo tutto il processo di addestramento sulla GPU. Questo permette una riduzione significativa dei tempi di addestramento, soprattutto in contesti che richiedono simulazioni complesse o un gran numero di agenti simultaneamente. Isaac Sim è altamente personalizzabile e può essere integrato con framework come PyTorch, rendendolo una scelta ideale per simulazioni complesse su larga scala. Tuttavia, uno dei principali svantaggi è rappresentato dai requisiti hardware, in quanto richiede prestazioni molto elevate per funzionare in modo efficiente, il che potrebbe non essere accessibile a tutti i team o progetti [78].

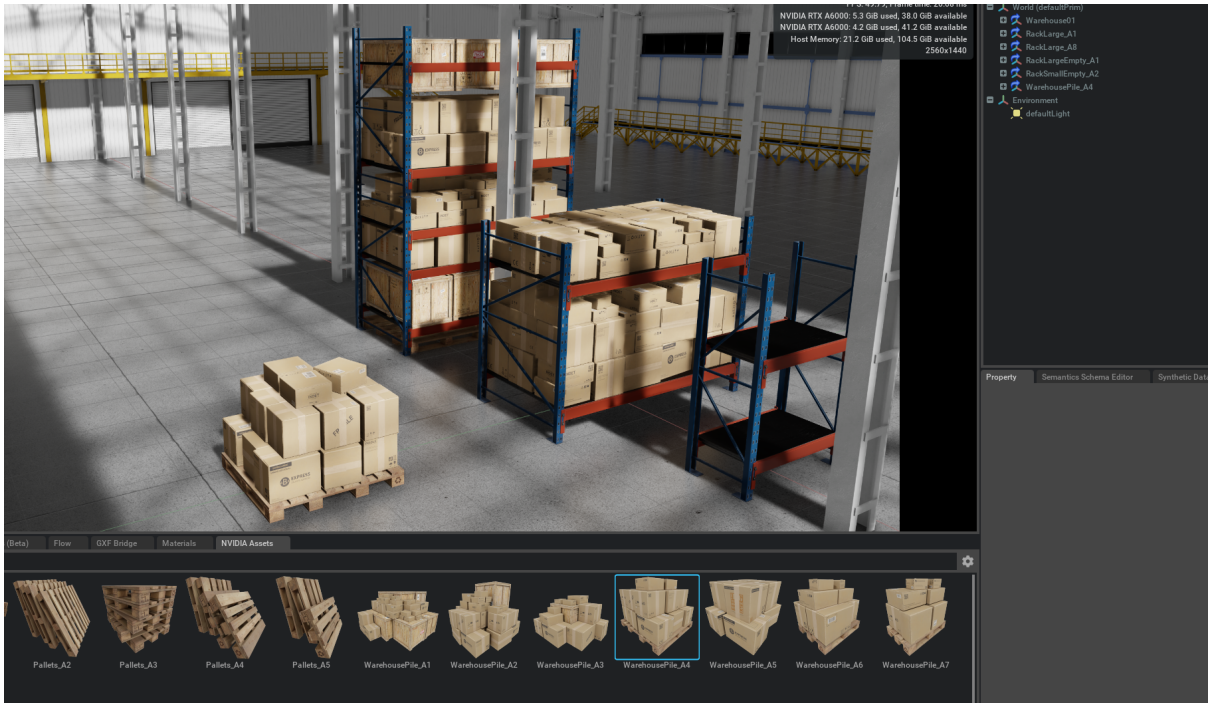

Figura 4.6: Esempio di ambiente di magazzino sviluppato con NVIDIA Omniverse [78].

4.3.1 Confronto tra i vari ambienti di simuazione

La valutazione è stata fatta mettendo a confronto i vari tool di OpenAI, Amazon e NVIDIA, considerando diversi aspetti di questi strumenti, come riportato nella tabella 4.2.

La scelta dell'ambiente di simulazione, dunque, dipende dalle necessità del progetto e dalle risorse disponibile. Il modello proposto, in questa fase di sviluppo, essendo una versione semplificata, che tiene in considerazione solo alcuni dei parametri possibili, non necessita di un ambiente di simulazione. Tuttavia, si ritiene che sviluppando modelli più complessi sarà necessario l'implementazione di uno di questi tool. Gym è la scelta più pratica e sufficiente per supportare il training delle prossime versioni del modello, che si concentrano su parametri semplificati. Nel momento in cui saranno richieste simulazioni più realistiche e l'inclusione di parametri aggiuntivi, potrebbe essere necessario passare a soluzioni più avanzate come AWS RoboMaker o NVIDIA Omniverse Isaac Sim. Questi strumenti offrirebbero un ambiente di simulazione più sofisticato e scalabile, particolarmente utile per testare scenari più complessi e realistici, in grado di riflettere meglio le dinamiche di un ambiente di magazzino reale.

	NVIDIA Isaac Sim	**AWS RoboMaker**	**GYM OpenAI**
Offerta	Simulazioni fisiche veloci, alta scalabilità, personalizzabile, pytorch ROS/ROS 2.0	Simulazioni cloud scalabili, supporto ROS, configurazione, deployment e scalabilità automatizzati, strumenti di visualizzazione integrati	Ambiente flessibile e personalizzabile, open source, compatibilità con librerie RL, simulazioni specifiche per compiti
Competenze	Python, C++, GPU e CUDA (utile), RL e ambienti di simulazione	Python, ROS, AWS, RL e pipeline CI/CD	Python, OpenAI Gym, RL, visualizzazione
Risorse	GPU prestanti, software compatile con pytorch e cuda	Infrastruttura cloud AWS, servizi AWS (EC2, S3, SageMaker), modello pay asyou-go	PC standard, Python e librerie correlate, risorse di calcolo moderate
Competenze di 3D design	Competenze di base necessarie	Competenze minime	Competenze minime
Tempo iniziale di setup	Alcune settimane	Alcune settimane a 1 mese	Pochi giorni a 1 settimana
Tempo per creazione magazzino	Alcune settimane a 2 mesi	Alcune settimane a 2 mesi	1 settimana a 1 mese
Tempo per test e integrazione	Alcune settimane	Alcune settimane	Alcune settimane

Tabella 4.2: Confronto tra gli applicativi per gli ambienti di simulazione.

Capitolo 5

Smart waving: sviluppo dell'algoritmo

Lo Smart Waving rappresenta una soluzione innovativa al problema del picking, una delle attività più onerose all'interno del magazzino, con un impatto significativo sui costi operativi e sull'efficienza della supply chain. Questa metodologia punta a ridurre drasticamente le inefficienze legate alla pianificazione dei task di prelievo, ottimizzando i percorsi degli operatori e diminuendo i tempi di inattività.
Le fasi di progettazione, l'analisi di fattibilità e la valutazione dei vari ambienti di simulazione hanno permesso di definire gli elementi costitutivi del modello, inclusi lo *action space*, l'*observation space* e la funzione di ricompensa. Tali elementi rappresentano la base per procedere al passaggio successivo, ossia l'implementazione dell'algoritmo vero e proprio.
Uno degli aspetti più innovativi di questo progetto risiede nell'integrazione del *reinforcement learning*, che consente all'algoritmo di apprendere dinamicamente le decisioni ottimali attraverso interazioni continue con l'ambiente simulato. Rispetto alle metodologie tradizionali, spesso basate su euristiche statiche o modelli deterministici, il reinforcement learning si adatta in tempo reale a cambiamenti nelle condizioni operative, garantendo una maggiore flessibilità e precisione.
La fase di sviluppo dell'algoritmo ha richiesto competenze avanzate di programmazione, in particolare l'uso di Python, uno dei linguaggi più utilizzati nel campo del *machine learning*. Grazie alla collaborazione con il team *Data Science* di Logistics Reply s.r.l., è stato possibile implementare e ottimizzare il modello, garantendo l'integrazione fluida con i sistemi esistenti e migliorando le prestazioni complessive del sistema di picking.

Nel corso dell'analisi, sono stati presi in cosiderazione due diversi approcci al problema, con complessità e prestazioni differenti. La valutazione di queste soluzioni è stata effettuata tenendo in considerazione diversi aspetti, tra cui l'efficienza computazionale, la

semplicità di implementazione, la capacità di generalizzare in scenari operativi diversi, e l'accuratezza dei risultati ottenuti. A seguito di un confronto approfondito, si è optato per l'approccio che, pur garantendo performance accettabili, richiedesse un livello di complessità di sviluppo sostenibile, evitando di sovraccaricare il sistema con eccessive risorse computazionali o tempi di sviluppo troppo lunghi. Inoltre, essendo questa una versione iniziale del modello, si è scelto di concentrarsi su un'architettura che bilanciasse prestazioni e facilità di sviluppo. In questo modo, è stato possibile implementare un algoritmo funzionante che possa servire come base per futuri miglioramenti e ottimizzazioni. Con l'evoluzione delle tecnologie e l'acquisizione di dati aggiuntivi, sarà possibile affinare ulteriormente il modello per migliorarne l'efficienza e l'efficacia, senza compromettere la semplicità del framework di base.

Infine, è stata definita la procedura per la fase di training, includendo i parametri tecnici su cui agire, la durata del periodo di addestramento, il dataset su cui svolgere tale attività e il numero di step realizzabili per ciascun episodio. Un'accurata definizione di questi elementi è essenziale per garantire un corretto processo di apprendimento del modello, in modo da bilanciare la necessità di ottenere risultati accurati e la sostenibilità computazionale della procedura. Ad esempio, definire un numero di episodi sufficiente a garantire che il modello raggiungesse un livello accettabile di convergenza, evitando però di incorrere in un overfitting del modello, oppure stabilire aggiornamenti consistenti della policy, per mantenere stabilità nel corso del training. Un ulteriore apetto da considerare, è il bilanciamento tra *exploration* e *exploitation*, anch'esso regolato tramite un fine-tuning dei vari fattori. Durante le prime fasi di addestramento, un'elevata esplorazione consente all'agente di scoprire nuove strategie, mentre nelle fasi finali è importante favorire lo sfruttamento delle politiche già apprese.

L'attività di tuning ha richiesto particolare attenzione, in quanto parametri troppo aggressivi avrebbero potuto condurre a una convergenza precoce su soluzioni subottimali, mentre un tuning troppo conservativo avrebbe rallentato eccessivamente il processo di apprendimento. È stato quindi necessario trovare un compromesso per garantire l'efficacia e la stabilità del modello anche nei contesti operativi più variabili.

5.1 Approcci al problema considerati

L'analisi del problema ha richiesto particolare attenzione, valutando i diversi approcci possibili, restringendo il campo a due differenti modalità operative, diverse per complessità e caratteristiche tecniche. Un primo approccio considerava la creazione di N cluster di pari dimensioni a partire da un set iniziale di ordini. La seconda metodologia valutava la realizzazione di un singolo cluster della dimensione desiderata a partire dalla medesima lista di ordini disponibili. La valutazione delle prestazioni è stata fatta per entrambe le

tecniche, tenendo conto degli elementi precedentemente citati. In questa sezione vengono analizzati in dettaglio entrambi gli approcci, considerando i vantaggi, gli svantaggi e la loro applicabilità al problema del picking.

5.1.1 Primo approccio: creazione di N cluster in contemporanea

Il primo approccio valutato prevede la realizzazione di N cluster in contemporanea, a partire da un set di ordini. Il numero di raggruppamenti creati dipende dalla quantità di ordini presenti all'interno della lista e dal vincolo legato alle dimensioni di ciascun cluster. L'agente, in questo scenario, ha la possibilità di operare effettuando scambi di ordini tra due cluster o spostando un ordine in un raggruppamento diverso da quello di appartenenza. In aggiunta, questo approccio richiede la realizzazione di un *pre-clustering*, al fine di creare i task di partenza, ordinati in base alla data di spedizione. A ogni iterazione, il modello seleziona un ordine da uno dei task e lo sposta in un cluster differente, con l'obiettivo di massimizzare la propria funzione di reward.
La figura 5.1 mostra uno schema illustrativo del funzionamento del modello.

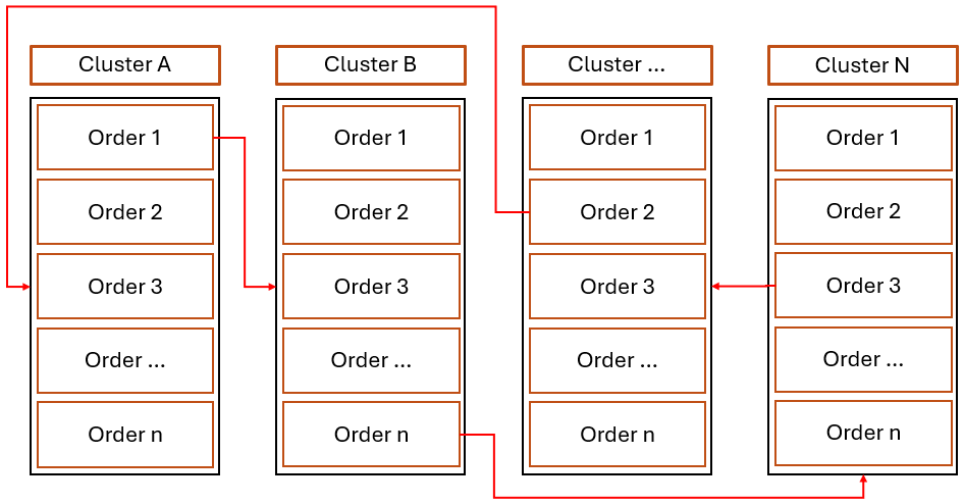

Figura 5.1: Schema di rappresentazione del primo approccio valutato.

Il principale beneficio derivante dall'implementazione di questa metodologia risiede nel bilanciamento tra i task. Infatti, ogni cluster risultante sarà di pari dimensioni, facilitando una distribuzione equa del carico di lavoro. Tuttavia, esistono alcune limitazioni significative all'implementazione di tale approccio.
In primo luogo, per poter applicare un planning aggregato, è necessario che tutti gli ordini da pianificare siano disponibili simultaneamente, il che non è sempre possibile. Nella maggior parte dei contesti aziendali, infatti, gli ordini di picking vengono inviati a blocchi, in base alle esigenze. Un ulteriore problema riguarda la pianificazione anticipata di ordi-

ni con data di spedizione futura. Questi ordini, se pianificati troppo presto, potrebbero occupare inutilmente le aree destinate alla spedizione, riducendo lo spazio disponibile per ordini più urgenti con scadenze di spedizione precedenti.
Per quanto riguarda lo sviluppo dell'algoritmo, la creazione simultanea di più cluster richiede uno sforzo superiore e una complessità computazionale elevata. L'agente, infatti, dovrebbe eseguire un numero significativamente maggiore di azioni per ciascun episodio, incrementando esponenzialmente il tempo richiesto per il training del modello. Durante i test condotti su questo approccio, si è osservato che l'agente non riusciva a convergere a una soluzione accettabile, nonostante fossero stati eseguiti 10 milioni di step. Questo risultato sottolinea come l'elevato numero di azioni richieste per gestire i N cluster porti a un rallentamento del processo di apprendimento, compromettendo l'efficienza complessiva. Nonostante gli sforzi fatti per ottimizzare il processo di training, l'elevato numero di operazioni richieste dall'agente per gestire N cluster ha determinato un rallentamento significativo nel processo di apprendimento. Durante i test, l'agente mostrava miglioramenti iniziali ma tendeva a convergere verso soluzioni subottimali, con una funzione di reward che non raggiungeva valori sufficienti a garantire un'efficace ottimizzazione del processo di picking. Questo comportamento è attribuibile alla complessità intrinseca del modello, che rendeva difficile per l'agente esplorare tutte le possibili configurazioni di cluster in modo efficiente.

Le difficoltà riscontrate nello sviluppo di tale approccio, in particolare l'elevata complessità computazionale e il mancato raggiungimento di una soluzione accettabile nonostante l'ampio numero di step, hanno portato alla necessità di valutare scenari differenti. L'obiettivo era ridurre il tempo di calcolo e la complessità del modello, pur mantenendo buone performance. Per raggiungere questo risultato, si è resa necessaria l'esplorazione di approcci alternativi, che consentissero una maggiore flessibilità operativa e una gestione più dinamica degli ordini.

5.1.2 Secondo approccio: creazione di un singolo cluster

Il secondo approccio valutato ha portato a risultati estremamente soddisfacenti, superando le criticità riscontrate nella metodologia precedentemente analizzata. Questo metodolo prevede la creazione di un singolo cluster a partire dalla pool iniziale di ordini. L'agente, in questo scenario, dispone di un set di azioni ridotto, ma più facilmente ripetibili in contesti diversi: selezionare un ordine dalla lista e spostarlo all'interno del *box*. Di conseguenza, a ogni iterazione, l'algoritmo identifica l'ordine "migliore" da inserire nel box, con l'obiettivo di massimizzare la propria di reward, rispettando i vincoli imposti. Un'azione potrà essere considerata *invalida*, qualora venga selezionato un ordine precedentemente inserito all'interno del box.

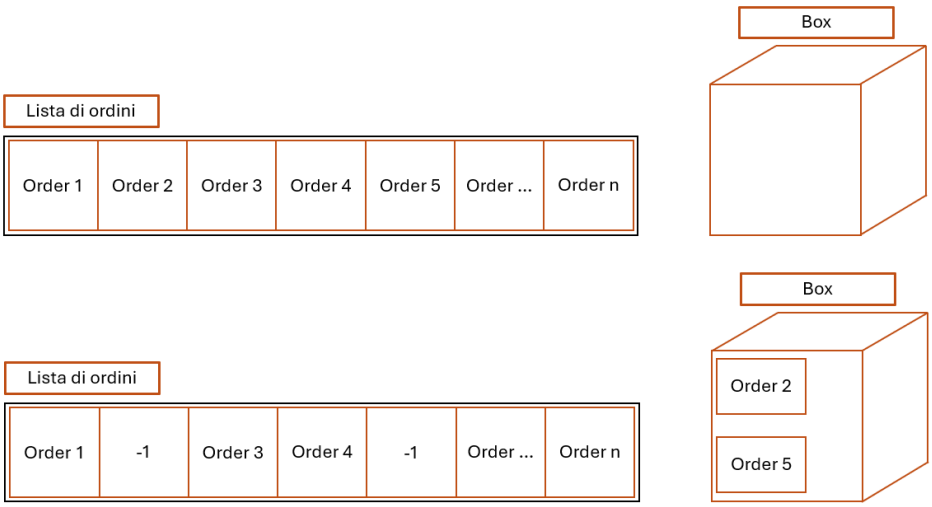

Figura 5.2: Schema di rappresentazione del secondo approccio valutato.

La principale limitazione all'implementazione di questo approccio è legata al potenziale scarso bilanciamento tra i task creati, dovuto alla pianificazione separata di ognuno di essi. Tuttavia, questo svantaggio è meno rilevante nel contesto operativo reale, poiché le aziende tendono a gestire ordini in modo continuo e dinamico. I benefici derivanti dall'introduzione di questa metodologia sono numerosi e superano di gran lunga questa criticità.
In primo luogo, la pianificazione di un singolo task consente di allinearsi al flusso operativo della gran parte delle aziende: ogni qualvolta un quantitativo di ordine viene reso disponibile, il sistema valuta se questi sono adatti alla creazione di un task ottimale. Qualora non lo siano, e le *due date* lo permettano, l'algoritmo rimarrà in attesa della *wave* successiva prima di procedere alla pianificazione. Questo meccanismo consente all'algoritmo di individuare l'istante ottimale per generare un picking task, evitando la creazione di cluster subottimali e aumentando l'efficienza complessiva del processo di picking.
l'eliminazione della necessità di un tool per il pre-clustering degli ordini riduce ulteriormente la complessità, valorizzando appieno la flessibilità tipica degli algoritmi di *reinforcement learning*.
Per quanto riguarda lo sviluppo dell'algoritmo, lo sforzo necessario e la complessità computazionale sono significativamente ridotti. Il numero di azioni eseguite dall'agente è diminuito drasticamente e dipende dalla dimensione del cluster che si vuole realizzare. Nonostante la semplificazione, le prestazioni, come illustrato nel capitolo successivo, sono decisamente superiori rispetto all'approccio precedente, garantendo una migliore efficienza operativa e un tempo di calcolo inferiore.

Le fasi successive di sviluppo hanno, dunque, coinvolto l'implementazione di questo secondo approccio, elaborando un tool in grado di realizzare un singolo cluster ottimale a

partire da un set di ordini, identificando l'istante migliore per procedere alla pianificazione di un task di picking.

5.2 Implementazione del codice di programmazione

L'implementazione del codice è stata la fase centrale dello sviluppo dell'algoritmo, in cui sono stati tradotti in linguaggio python i concetti descritti durante la progettazione del modello, come la reward function, l'action space e l'observation space. L'obiettivo principale è stato creare un'architettura efficiente che permettesse all'agente di interagire con l'ambiente di simulazione, apprendendo strategie ottimali per la risoluzione del task di picking.

Per lo sviluppo dell'algoritmo, è stato utilizzato l'ambiente *gym* di OpenAI, una piattaforma ampiamente adottata per la creazione di ambienti di reinforcement learning. Questa scelta si è rivelata vantaggiosa per la disponibilità di strumenti utili alla creazione e alla gestione dell'ambiente di simulazione, facilitando l'implementazione e il testing dell'algoritmo.

5.2.1 Libreria gym: una breve panoramica

Lo sviluppo del codice ha richiesto l'utilizzo della libreria *Gym*, implementata da OpenAI, e in particolare dell'ambiente *Gymnasium*. Gymnasium fornisce un'*application programming interface (API)*, adatta alla configurazione di ambienti per il training di algoritmi di reinforcement learning. Attualmente, l'applicazione di questa piattaforma è ampiamente diffusa nell'implementazione di videogiochi (es. cartpole, atari, etc.). Il core di tale applicativo è una classe di python ad alto livello, rappresentante un *processo decisionale di markov (MDP)*.

Un MDP è costituito da un set di stati e uno di azioni, una funzione di reward e una funzione di transizione; quest'ultima determina, stocasticamente, lo stato successivo dell'ambiente come funzione dello stato corrente e dell'azione eseguita. La funzione di reward stabilisce la ricompensa ricevuta dall'agente. In generale, un MDP può essere caratterizzato da spazi di azioni e osservazioni di dimensioni infinite, tuttavia, nel caso in analisi verranno considerati problemi discreti a dimensione finita [12].

Gymnasium consente di generare uno stato iniziale, muoversi a quello successivo e visualizzare l'ambiente. Gym offre alcune funzionalità chiave che sono state fondamentali per lo sviluppo del nostro ambiente di simulazione. In particolare, l'*action space* e l'*observation space* sono stati definiti con facilità utilizzando le classi predefinite di Gym, che permettono di specificare lo spazio delle azioni e delle osservazioni come oggetti discreti o continui. La modularità della libreria consente anche di gestire gli episodi, monitorare il

comportamento dell'agente e visualizzare i progressi del training.
Una delle caratteristiche principali che ha reso Gymnasium ideale per questo progetto è la sua compatibilità con altre librerie di machine learning, come *TensorFlow* e *PyTorch*, che sono state utilizzate per l'implementazione del modello di apprendimento. La struttura ad alto livello e la semplicità di utilizzo della libreria hanno permesso di focalizzarsi sugli aspetti di ottimizzazione del training dell'agente, riducendo la complessità legata alla gestione dell'ambiente simulato.

Entrando nel dettaglio del codice utilizzato, un ambiente viene creato e inizializzato tramite il comando `make()`, per poi procedere alla registrazione della prima osservazione con `Env.reset()`. Successivamente l'agente esegue un'azione mediante `Env.step()`, influenzando l'ambiente e ricevendo nuove osservazioni e la ricompensa relativa. Questa sequenza di scambi azione-osservazione è denominato *timestep*. Il ciclo tipico di un *timestep* può essere rappresentato come segue:

```
done = False
obs = env.reset()
while not done:
    action = agent.act(obs)  # L'agente seleziona un'azione
    obs, reward, done, info = env.step(action)  # Esegue l'azione e
        ottiene nuove osservazioni
    agent.learn(obs, reward)  # L'agente aggiorna la propria strategia
```

Listing 5.1: Esempio di ciclo di timestep in Python

La chiusura dell'environment può avvenire in vari modi: nel momento in cui l'agente completa il task, oppure quando viene raggiunto il numero prefissato di episodi. Per quanto riguarda gli spazi delle azioni e delle osservazioni, questi vengono definiti all'interno degli attributi `action_space` e `observation_space`. Data l'importanza delle osservazioni, è conveniente disporre di un metodo per tradurre gli stati dell'ambiente in leggibili dall'agente. Questo processo viene gestito tramite `_get_obs(self)` , che restituisce lo stato attuale dell'ambiente in una forma utile per la decisione successiva dell'agente [79].

5.2.2 Analisi tecnica del codice di programmazione

L'implementazione del codice è stata articolata in diverse fasi, che hanno coinvolto la creazione dell'ambiente, l'elaborazione dell'input fornito, il calcolo delle metriche necessarie alla valutazione delle performance dell'algoritmo, la definizione della reward function e la definizione di funzioni plotting per visualizzare l'andamento di vari parametri nel tempo, che evidenziano il processo di apprendimento dell'agente durante il periodo di training.

L'input fornito al modello è una lista di ordini in formato *json*, contenente informazioni riguardanti il codice identificativo, la *due date*, l'ubicazione di ciascun articolo presente

nelle righe d'ordine, il peso e il volume complessivi, e il *picking path*, che rappresenta la distanza tra la postazione di picking e l'ubicazione dell'articolo nel magazzino, come illustrato nell'esempio seguente. I valori delle varie voci non vengono riportati per questioni di privacy.

```
{
  "orders": [
    {
      "id": "0",
      "locations": [
        {
          "aisleGroup": {
            "code": "GV"
          },
          "mapLocationLabel": null,
          "aisle": **,
          "span": **,
          "spanPosition": *,
          "level": *,
          "pickingPath": ******
        },
        {
          "aisleGroup": {
            "code": "GV"
          },
          "mapLocationLabel": null,
          "aisle": **,
          "span": **,
          "spanPosition": *,
          "level": *,
          "pickingPath": ******
        }
      ],
      "volume": **,
      "weight": **,
      "priority": false,
      "dueDate": **********,
      "quantity": *
    }
  ]
}
```

Listing 5.2: Esempio di ordine fornito in input.

L'ambiente è stato realizzato tramite l'utlizzo della libreria gym, precedentemente descritta.

```python
class WarehouseEnv(gym.Env):
    def __init__(self, orders, max_episode_steps, max_orders=10,
        max_volume=35, max_weight=25):
        self.orders = orders
        self.observation_space = spaces.Box(low=0,  high=np.inf, shape
            =(1,6), dtype=np.float32)
        self.current_episode_step = 0
        self.current_step = 0
        self.max_round_steps = max_episode_steps
        self.max_orders = max_orders
        self.max_volume = max_volume
        self.max_weight = max_weight
        self.action_list = list(range(0, len(self.orders)))
        self.action_space = spaces.Discrete(len(self.action_list))
        self.order_id_list = generate_orders_id_array(self.orders)
        self.box = []
        # Configure logging
        self.logger = logging.getLogger(__name__)
        logging.basicConfig(level=logging.INFO, format='%(asctime)s -
            %(levelname)s - %(message)s')
        self.box_metrics = None
        self.box_orders_list = []
```

Listing 5.3: Creazione dell'ambiente di training in python.

La funzione sopra descritta crea l'ambiente simulato del magazzino, inizializzando le variabili globali necessarie per l'esecuzione dell'algoritmo. La classe definisce il comportamento dell'ambiente e configura il registro (*logger*) per il monitoraggio degli eventi e degli stati. All'interno della classe così creata esistono, in aggiunta, diverse funzioni che definiscono il comportamento dell'agente nel contesto.

Una delle funzioni di maggiore rilevanza è `step()`, che regola il processo di azione dell'agente. Questa funzione viene eseguita ogni volta che l'agente esegue un'azione, spostando uno degli ordini nel *box*. Successivamente, l'impatto dell'azione viene valutato, analizzando le osservazioni ricevute e la ricompensa.

```python
    def step(self, action):
        if action >= self.action_space.n:
            raise ValueError("Invalid action")

        self.current_episode_step += 1
        self.current_step += 1

        #in None in the case of the orders was olready selected (i.e.,
           it is in the boc)
        selected_order = self.get_order_by_id(self.orders, self.
           order_id_list[action])

        if selected_order is None:
            reward = 0
            self.logger.info("Invalid Action")

        else:
            self.box.append(selected_order)
            self.order_id_list[action] = -1
            reward = self.calculate_reward()

        observation = self._get_observation()

        # Check termination conditions
        done = self.check_termination_condition()

        return observation, reward, done, {}
```

Listing 5.4: Funzione `step()`..

La funzione `reset()` viene chiamata all'inizio di ogni nuovo episodio per re-inizializzare lo stato dell'ambiente e ripristinare i contatori interni. Questo garantisce che l'agente riparta da zero per ogni ciclo di training.

```python
    def reset(self):
        super().reset(seed=None)
        self.box = []
        self.order_id_list = generate_orders_id_array(self.orders)
        self.invalid_actions = 0
        self.current_episode_step = 0
        self.logger.info("Environment reset. Clusters reinitialized.")
        return self._get_observation()
```

Listing 5.5: Funzione `reset()`.

Il calcolo della ricompensa è una componente chiave per guidare l'apprendimento dell'agente. La funzione `calculate_reward()`calcola la ricompensa basata su vari parametri, come il volume e il peso del *box*, e normalizza il valore ottenuto per facilitare l'apprendimento dell'agente.

```
    def calculate_reward(self):
        if len(self.box) == 0:
            return 0

        #Box metrics
        box_size,total_weight,total_volume,std_due_dates,std_aisle,
            std_span = extract_box_metrics(self.box)
        max_due_date_std, max_aisle_std, max_span_std =
            calculate_max_values(self.orders)

        # Coefficients
        a = 4  # Higher weight for due date standard deviation
        b = 1  # Weight for span standard deviation (now not relevant)
        c = 1  # Weight for aisle standard deviation
        d = 3  # Weight for volume closeness to max
        e = 3  # Weight for weight closeness to max
        f = 4  # Weight for cluster size closeness to max

        # Quadratic penalty rewards
        RDEVSTD_D_i = a * (1 - np.sqrt(std_due_dates / max_due_date_std
            ))
        RDEVSTD_S_i = b * (1 - np.sqrt(std_span / max_span_std))
        RDEVSTD_A_i = c * (1 - np.sqrt(std_aisle / max_aisle_std))

        # Asymmetric penalty for exceeding volume and weight
        RV_i = -1 * d if total_volume > self.max_volume else d *
            total_volume/self.max_volume
        RP_i = -1 * e if total_weight > self.max_weight else e *
            total_weight/self.max_weight
        RcSIZE_i = -1 * f if box_size > self.max_orders else f *
            box_size/self.max_orders

        Rmin = -10
        Rmax = 16
        total_reward =  RDEVSTD_D_i + RDEVSTD_S_i + RDEVSTD_A_i + RV_i
            + RP_i + RcSIZE_i
        norm_reward = (total_reward - Rmin) / (Rmax - Rmin)
        return norm_reward
```

Listing 5.6: Funzione per il calcolo della ricompensa.

La funzione `_get_observation()` estrae le metriche rilevanti dal *box*, permettendo all'agente di prendere decisioni informate. La raccolta delle informazioni essenziale avviene tramite `extract_box_metrics()`, che si occupa di acquisire i dati necessari dall'input fornito al modello, rendendolo leggibile e modificabile dall'algoritmo. Questi dati sono fondamentali per l'agente, poiché lo supportano nella scelta riguardo alla prossima azione da intraprendere durante il training.

```
    def _get_observation(self):
        return extract_box_metrics(self.box)

    def extract_box_metrics(box):

        if len(box) == 0:
            return 0,0,0,0,0,0

        # Extract cluster metrics
        box_size = len(box)
        total_weight = round(sum(order['weight'] for order in box),1)
        total_volume = round(sum(order['volume'] for order in box),1)

        due_dates = [order['dueDate'] for order in box]
        std_due_dates = round(float(np.std(due_dates))/3600,1)

        aisle_positions = [location['aisle'] for order in box for
            location in order['locations']]
        std_aisle = round(float(np.std(aisle_positions)),1)

        span_positions = [location['span'] for order in box for
            location in order['locations']]
        std_span = round(float(np.std(span_positions)),1)

        return box_size,total_weight,total_volume,std_due_dates,
            std_aisle,std_span
```

Listing 5.7: Funzione `_get_observations()`.

Un ulteriore porzione di codice estremamente rilevante sono le funzione racchiuse all'interno del `main()`, dove, a seguito dell'importazione dell'ambiente creato, vengono definiti parametri tecnici dell'algoritmo, che regolano il processo di aggiornamento della policy e l'apprendimento del modello, e varie funzioni, che intervengono nelle due fasi principali: il *training* e il *testing*.

Il monitoraggio della periodo di addestramento del modello viene fatto tramite la classe `TrainingMonitorCallback()`, il cui scopo è tenere traccia di metriche chiave e registrare informazioni sui vari episodi.

```python
class TrainingMonitorCallback(BaseCallback):
    def __init__(self, verbose=0):
        super(TrainingMonitorCallback, self).__init__(verbose)
        ...
        self.episode_box_sixe = []
        self.episode_box_weight = []
        self.episode_box_volume = []
        self.episode_box_std_data = []
        self.episode_box_std_aisle = []
        self.episode_box_std_span = []

        self.value_loss = []
        self.entropy_loss = []
        self.policy_loss = []
        self.approx_kl = []
        self.timesteps = []

    def _on_step(self) -> bool:
        reward = self.locals['rewards'][0]
        done = self.locals['dones'][0]

        self.current_rewards.append(reward)
        self.current_lengths += 1
        env = self.training_env.envs[0]

        if done:
            box_metrics = env.get_box_metrics()
            self.episode_box_sixe.append(box_metrics[0])
            self.episode_box_weight.append(box_metrics[1])
            self.episode_box_volume.append(box_metrics[2])
            self.episode_box_std_data.append(box_metrics[3])
            self.episode_box_std_aisle.append(box_metrics[4])
            self.episode_box_std_span.append(box_metrics[5])
            episode_reward = sum(self.current_rewards)
            self.episode_rewards.append(episode_reward)
            print("Episode " + str(self.current_episode) + " reward: "
                + str(episode_reward))
            self.episode_lengths.append(self.current_lengths)
            self.current_rewards = []
            self.current_lengths = 0
            self.current_episode += 1

        return True
    ...
```

Listing 5.8: Classe `TrainingMonitorCallback()`.

La funzione `__init__()` inizializza le variabili necessarie per tenere traccia delle metriche relative al box (dimensione, peso, volume, ecc.) e delle perdite del modello durante il training. La funzione `_on_step()` raccoglie le ricompense e la lunghezza dell'episodio, registrando tali valori. Le variabili monitorate includono:

- **Box size**: il numero di ordini contenuti nel box al termine di ogni episodio.
- **Box weight**: il peso totale degli ordini presenti nel box.
- **Box volume**: il volume complessivo degli ordini nel box.
- **Standard deviation of due dates, aisles, e spans**: queste misurazioni sono importanti per valutare la distribuzione degli ordini in base a data di scadenza, corsie e campate, e quindi indicano la coerenza e l'efficienza della selezione degli ordini.

Queste metriche vengono utilizzate per valutare la performance dell'algoritmo durante il periodo di addestramento e per assicurarsi che l'agente stia apprendendo strategie efficaci.

Oltre alla registrazione delle metriche, la classe `TrainingMonitorCallback()` genera grafici che rappresentano l'andamento dei vari parametri durante il periodo di training. Questi grafici includono, ad esempio, l'evoluzione della ricompensa accumulata per episodio e l'andamento delle deviazioni standard dei vari parametri misurati (peso, volume, date di consegna, corsie, ecc.). La visualizzazione grafica di questi parametri aiuta a comprendere come l'agente sta migliorando nel tempo e se ci sono segnali di overfitting o stagnazione.

All'interno del main si può osservare l'intero procedimento seguito dal modello, dalla creazione dell'ambiente fino alla fase di testing. La configurazione della fase di training prevede la definizione dei valori di riferimenti per le metriche in analisi, della policy utilizzata e dei parametri tecnici che regolano il processo Oltre al fine-tuning delle specifiche, è essenziale definire il numero di *timesteps* complessivi. Il periodo di training dovrà essere sufficientemente lungo, in modo da consentire all'agente una prima fase di esplorazione per poi ripetere le azioni che portano a una più elevata reward, senza, però, incorrere nell'overfitting.

```
if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(
        levelname)s - %(message)s')
    training_monitor_callback = TrainingMonitorCallback()
    max_episode_steps = 15
    max_invalid_actions = 3
    max_orders = 10
    max_volume = 40
    max_weight = 25
    # Create vectorized environment
    env = make_vec_env(lambda: make_env(max_episode_steps, max_orders,
        max_volume, max_weight), n_envs=1)
    # Extract  box configuration
    out_box = []
    for idx in range(env.num_envs):
        out_box.append(env.envs[idx].box.copy())
    # Define and train the RL agent
    model = PPO(
        'MlpPolicy',
        env=env,
        n_epochs=30,
        max_grad_norm = 0.5,
        batch_size=512,
        learning_rate=1e-5,
        gamma=0.99,
        ent_coef = 0.002,
        gae_lambda=0.94,
        clip_range=0.03,
        n_steps=2048,
        verbose=1
    )
    # total_timesteps specifies the total number of steps the agent
        will take in the environment
    model.learn(total_timesteps=4500000, callback=[
        training_monitor_callback])
    # Evaluate and visualize agent performance
    obs = env.reset()
    episode_rewards = []
    final_box_configurations = []  # To store final box metrics after
        each episode
    final_box_list = []
```

Listing 5.9: Fase di training all'interno del `main()`.

Una volta completato il training, è fondamentale testare le performance dell'agente su nuovi episodi, valutando se è in grado di applicare efficacemente le strategie apprese.

La fase di testing permette di verificare come l'agente si comporta su dati non visti e di analizzare la qualità delle decisioni prese. La configurazione di tale processo consiste nel riprodurre quanto appreso durante il training, senza la necessità di registrare le osservazioni al termine di ogni episodio.

```
print("\n\n\n\n TESTING")
    max_reward = 0
    best_box_metrics = []
    best_box_list = []
    for _ in range(10):
        # Reset the observation for a new episode
        obs = env.reset()
        done = False
        episode_reward = 0

        while not done:
            action, _states = model.predict(obs)
            obs, rewards, done, info = env.step(action)
            env.render()
            episode_rewards.append(rewards)
            episode_reward += rewards[0]  # Accumulate rewards for the
                episode

        # After the episode ends, get the final box metrics
        final_box_metrics = env.get_attr('box_metrics')[0]  # Fetch the
            box metrics
        final_box_order_list = env.get_attr('box_orders_list')[0]
        final_box_configurations.append(final_box_metrics)
        final_box_list.append(final_box_order_list)

        if episode_reward > max_reward:
            max_reward = episode_reward
            best_box_metrics = final_box_metrics
            best_box_order_list = final_box_order_list
```

Listing 5.10: Fase di testing all'interno del `main()`.

Conclusa l'intera procedura, l'output fornito dal modello è un cluster costituito da n ordini. L'algoritmo restituisce il codice identificativo di ciascuno di essi e i valori delle metriche per ciascun box creato nella fase di testing. Questi valori permettono di valutare la qualità del cluster generato e di comprendere come l'agente ha appreso e applicato le strategie durante il training.

5.3 Definizione della fase di training

La fase di sviluppo si è conclusa con la definizione del processo di addestramento del modello, individuando i valori limite che possono essere assunti dagli indicatori di performance, la durata del periodo di traning e i parametrici tecnici dell'algoritmo. Le decisioni prese hanno richiesto un'attenta analisi dell'input e delle esigenze dell'azienda, per identificare valori che fosse coerenti con il contesto reale.

Le misure massime raggiungibili dalle metriche considerate sono state determinate in base alle necessità aziendali, come indicato di seguito:

- Massimo numero di step per episodio: 15.
- Massimo numero di azioni invalide: 3
- Dimensione massima del box: 10 ordini.
- Volume massimo del box: 40 cm^3
- Peso massimo del box: 25 kg

Le scelte fatte sono state dettate dalle esigenze operative dell'azienda considerata nello studo, che utilizzava carrelli per il picking con un massimo di dieci ordini trasportabili per pick-tour. A partire da questo dato, e conoscendo il valore medio di peso e volume degli articoli, è stato possibile stabilire dei limiti accettabili per i parametri di volume e peso del box. Infine, dato che la capienza del box non può eccedere i dieci ordini, all'agente sono state consentite 15 azioni per episodio, con 3 azioni invalide. La decisione di ammettere fino a tre azioni invalide per episodio è stata presa in modo arbitrario, con l'aspettativa che, nel corso del training, il numero di errori commessi dall'agente diminuisca progressivamente, tendendo verso lo zero man mano che l'agente impara a ottimizzare le proprie scelte. In un momento successivo si è valutato di analizzare il comportamento del modello riducendo i valori di peso e volume, con l'obiettivo di complicare il processo di ottimizzazione e testare la robustezza dell'algoritmo in condizioni operative più sfidanti.

La decisione riguardo il numero di timesteps complessivi del periodo ti training è stata presa considerando una fase iniziale di validazione dell'algoritmo, di breve durata, in modo da verificare il corretto funzionamento del sistema e identificare eventuali errori o malfunzionamenti. A seguito di tale verifica, il numero di timesteps è stato progressivamente aumentato, cercando un equilibrio tra un tempo sufficiente a consentire una prima fase di esplorazione da parte dell'agente e il rischio di overfitting del modello. Questo approccio ha permesso di ottimizzare il processo di apprendimento senza compromettere la generalizzazione del modello.

Parallelamente alla definizione della durata della fase di training, è stato eseguito un processo di *fine-tuning* dei parametri tecnici, con l'obiettivo di garantire un corretto apprendimento da parte dell'agente e stabilità della policy negli aggiornamenti successivi alla conclusione di un episodio. I parametri considerati sono i seguenti:

- **Divergenza KL approssimata**: misura la divergenza tra la vecchia e la nuova policy a seguito di un aggiornamento. Indica quanto la policy sta cambiando durante l'addestramento. Idealmente, questo valore dovrebbe essere ridotto (vicino a 0 ma non esattamente 0). Nella *PPO*, la policy attualmente utilizzata, se questo parametro è eccessivamente elevato, indica che la policy sta cambiando troppo, suggerendo instabilità. Viceversa, un valore troppo basso potrebbe indicare che la policy non sta apprendendo efficacemente.

- **Clip Fraction**: la frazione degli aggiornamenti in cui il rapporto di probabilità tra la nuova e la vecchia policy è stato "tagliato" (*clipped*). Questa metrica indica quanto spesso il clipping dell'obiettivo di PPO influisce sugli aggiornamenti. Valori compresi tra 0.1 e 0.3 sono generalmente considerati ottimali. Un valore molto basso potrebbe suggerire che la policy non viene aggiornata a sufficienza, mentre un valore troppo alto potrebbe indicare che la policy sta subendo cambiamenti eccessivi.

- **Entropy Loss**: l'entropia misura la casualità nella distribuzione delle azioni; una maggiore valore di entropia implica un aumento dell'esplorazione. Un valore negativo moderato è spesso desiderabile. Se tale parametro è troppo basso (ovvero molto negativo), significa che la policy sta diventando deterministica, suggerendo una ridotta esplorazione e una possibile convergenza prematura a una policy subottimale.

- **Explained Variance**: questa metrica quantifica la porzione di varianza nel valore target spiegata dalla funzione di valore. È un indicatore della bontà di adattamento della funzione di valore ai rendimenti. Valori prossimi a 1.0 sono considerati ottimali (indicando previsioni della funzione di valore accurate), mentre valori vicini a 0 o negativi suggeriscono che la funzione di valore non sta apprendendo correttamente.

- **Policy Gradient Loss**: è la perdita associata all'aggiornamento del gradiente della policy. Riflette l'entità dello step nella riduzione del gradiente della policy. Solitamente, questo valore è negativo. Un perdita stabile e ridotta suggerisce stabilità.

- **Value Loss Plot**: una diminuzione della perdita di valore consistente indica che la value function sta migliorando, suggerendo un progresso positivo nell'apprendimento della policy. Improvvisi picchi o una perdita di valore elevata possono indicare instabilità o problemi con il tasso di apprendimento.

La definizione della fase di training ha rappresentato la conclusione dello sviluppo del modello. Il passaggio successivo ha riguardato il testing del modello e l'analisi delle prestazioni, confrontando i risultati ottenuti dall'integrazione del reinforcement learning con quelli del sistema tradizionale. Questo processo ha l'obiettivo di valutare i benefici derivanti dall'introduzione dello Smart waving nel processo di generazione dei picking task. Tale tematica verrà trattata nel prossimo capitolo.

Capitolo 6

Smart waving: fase di testing del progetto

Il codice implementato e i primi test di breve durata hanno confermato la corretta operatività del modello, senza evidenziare la presenza di errori o malfunzionamenti. La fase progettuale successiva è stata incentrata sul *fine-tuning* delle caratteristiche tecniche dell'algoritmo di reinforcement learning e sull'individuazione della durata ottimale del periodo di training.
Le principali sfide affrontate in questa fase includono: garantire un corretto processo di apprendimento del modello, evitando instabilità nella policy dovute a update eccessivamente aggressivi, e una crescita costante della funzione di reward nel tempo. Inoltre, è stato necessario individuare una lunghezza adeguata per la fase di addestramento del modello, trovando un giusto equilibrio tra una durata sufficiente a consentire l'esplorazione di diverse soluzioni da parte dell'agente e un arco di tempo eccessivo che porterebbe all'*overfitting* o stagnazione. Tali operazioni richiedono particolare attenzione, in quanto incidono profondamente sulle performance del modello, richiedendo quindi un impegno significativo nella loro configurazione.

La fase di testing è stata articolata su diversi scenari, con l'obiettivo di valutare le prestazioni dello Smart waving di fronte a diverse situazioni. La lista di ordini fornita in input è rimasta invariata, mentre sono state modificate le condizioni al contorno, tra cui il numero di ordini da pianificare e i limiti superiori delle variabili d'interesse. I test eseguiti hanno evidenziato i punti di forza del modello, ma anche le sue debolezze, mostrando la necessità di ulteriori miglioramenti per trasformare questo strumento, che già possiede un elevato potenziale, in un tool capace di incrementare notevolmente l'efficienza di qualsiasi magazzino. Le valutazioni sono state fatte ponendo a confronto i risultati ottenuti a seguito dell'integrazione dello Smart waving nel processo di pianificazione derivanti dall'utilizzo del metodo attualmente in uso. Nonostante questa rappresenti una versione preliminare

dello strumento, sono già osservabili significativi benefici, che verranno approfonditi in seguito.

La soluzione proposta in questo documento, come evidenziato più volte, è una prima versione di uno strumento che mira all'ottimizzazione della pianificazione dei task di picking. Tuttavia, sono presenti diverse limitazioni, tra cui una ridotta flessibilità e adattabilità a cambiamenti nel contesto. Gli sviluppi futuri, dunque, si concentreranno sul superamento di queste criticità, con l'obiettivo di creare un modello estremamente versatile e applicabile a contesti dinamici. Il principale vantaggio del reinforcement learning risiede nella sua polivalenza, che gli consente di essere impiegato in una vasta gamma di applicazioni all'interno della gestione del magazzino. Pertanto, lo Smart waving rappresenta solo un primo tentativo di integrazione di questa tecnologia all'interno del mondo della supply chain, con la possibilità di estenderne l'utilizzo all'intero processo di *outbound* e *inbound*, fino alla selezione dei fornitori e l'identificazione delle rotte ottimali nella last-mile logistics.

6.1 Confronto tra i risultati a monte e a valle dell'integrazione con il Cluster Optimizer

La valutazione delle prestazioni del sistema a valle dell'integrazione dello Smart waving con il *Cluster Optimizer* ha ulteriormente evidenziato le potenzialità di questo strumento. Il modulo, attualmente, prende in considerazione vari parametri di input, tra cui la disposizione delle ubicazioni in magazzino, le specifiche dei carrelli/contenitori, le strategie di allocazione, l'ottimizzazione dei percorsi di picking e altre caratteristiche del processo operativo. Successivamente, riorganizza i job di picking in un numero ottimale di task, riducendo al minimo la distanza complessiva percorsa dagli operatori.
La criticità principale di tale strumento, come discusso in precedenza, risiede nella fase di pianificazione dei task, dove lo Smart waving può apportare significativi miglioramenti, ottimizzando l'efficienza del processo di picking.
Il confronto delle prestazioni è stato condotto in una seconda fase del testing e dell'analisi, dopo aver effettuato un accurato *fine-tuning* dei parametri del modello. Questo passaggio è stato necessario per garantire un corretto apprendimento del modello e stabilità nella sua operatività.

6.1.1 Processo di fine-tuning dei parametri tecnici del modello

Il processo di fine-tuning dei parametri del modello ha richiesto un notevole impiego di tempo e risorse, con l'obiettivo di individuare i valori ottimali per gli indicatori descritti in precedenza, al fine di garantire un corretto *learning process* da parte dell'agente e mantenere la policy stabile a seguito degli aggiornamenti alla conclusione di ciascun

episodio. Questa fase ha comportato l'esecuzione di numerosi test di durata variabile, seguiti dalla valutazione dei grafici raffiguranti l'andamento delle metriche durante il training. A ogni iterazione, i parametri sono stati rivalutati, apportando leggere variazioni per identificare il punto di equilibrio ottimale.

I primi test hanno evidenziato le inefficienze presenti nel modello, che hanno richiesto ulteriori perfezionamenti per migliorare la stabilità e le performance complessive. I parametri inizialmente utilizzati sono stati i seguenti:

- $max\_grad\_norm = 0.3$ (*Norma massima del gradiente*): limita la norma dei gradienti durante l'addestramento per evitare aggiornamenti eccessivamente aggressivi che potrebbero destabilizzare l'apprendimento. É stato posto pari a 0.3, un valore relativamente ridotto, per garantire la stabilità del modello.
- $batch\_size = 128$ (*Dimensione del batch*): indica il numero di esempi di addestramento utilizzati per aggiornare i pesi del modello in ogni passo. Un batch size stabilito pari a 128 rappresenta una dimensione moderata, che bilancia l'efficienza del calcolo con la precisione degli aggiornamenti.
- $learning\_rate = 2e - 5$ (*learning rate*): determina la rapidità con cui il modello aggiorna i suoi pesi in risposta all'errore calcolato. Il tasso di apprendimento è stato fissato a 2e-5 (0,00002), particolarmente contenuto, il che significa che il learning process avviene lentamente e con maggiore precisione.
- $gamma = 0.99$ (*Fattore di sconto*): applicato alle ricompense future. Il valore di 0.99 indica che l'agente dà molto peso alle ricompense future, incentivando decisioni a lungo termine.
- $ent\_coef = 0.05$ (*Coefficiente di entropia*): indica quanto l'esplorazione viene incentivata nell'apprendimento. Un valore di 0.05 favorisce un livello moderato di esplorazione, mantenendo un certo grado di casualità nelle azioni intraprese dall'agente.
- $gae\_lambda = 0.85$ (Lambda del vantaggio generalizzato): è il fattore di smorzamento per il vantaggio generalizzato (*Generalized Advantage Estimation*). Un valore di 0.85 rappresenta un buon compromesso tra bias e varianza, migliorando la stabilità e la precisione della policy.
- $clip\_range = 0.3$ (*Intervallo di clipping*): definisce l'intervallo entro il quale la politica può essere aggiornata in PPO (*Proximal Policy Optimization*). Un valore di 0.3 consente una certa flessibilità nei cambiamenti della policy, pur mantenendo un controllo per evitare cambiamenti troppo bruschi.

- *n_steps* = 2048 (*Numero di passi per aggiornamento*): indica il numero di interazioni tra l'agente e l'ambiente prima di eseguire un aggiornamento della policy. Un valore di 2048 permette di raccogliere un ampio set di esperienze prima dell'aggiornamento, migliorando la stabilità dell'apprendimento.
- *Step_per_episodio* = 10 (*Numero di passi per episodio*): definisce il numero massimo di passi che l'agente può compiere all'interno di un singolo episodio. Questo limita la durata di un episodio e il numero di azioni che l'agente può intraprendere.
- *Step* = 1.000.000 (*Numero totale di passi*): numero totale di interazioni che l'agente esegue con l'ambiente durante l'intero processo di training. Un valore di 1 milione consente all'agente di esplorare molteplici strategie e ottimizzare le proprie decisioni in maniera completa.

I pesi assegnati alle variabili rilevanti nel processo sono stati mantenuti pressochè costanti, con alcune leggere variazioni in alcuni test, per valutarne l'effetto sulla modalità di apprendimento dell'agente:

- Dimensione del cluster: 4.
- Peso del cluster: 3.
- Volume del cluster: 3.
- Deviazione standard della due date all'interno del ragruppamento: 4.
- Deviazione standard del numero di corridoi all'interno del ragruppamento: 2
- Deviazione standard del numero di campate all'interno del ragruppamento: 2

I grafici ottenuti sono riportati di seguito. La funzione di reward è crescente, conforme alle aspettative, il che indica che l'agente sta apprendendo correttamente. Tuttavia, si osservano diversi picchi durante la fase di training, che suggeriscono una certa instabilità negli aggiornamenti della policy, in particolare durante i periodi di esplorazione. Questo potrebbe essere dovuto a un fine-tuning insufficiente di alcuni parametri, come il tasso di apprendimento (*learning rate*) o il fattore di entropia, che potrebbe incentivare un'esplorazione troppo aggressiva. Inoltre, è evidente che il valore della reward potrebbe essere ulteriormente migliorato prolungando la durata del training, offrendo all'agente più tempo per stabilizzare il proprio comportamento.

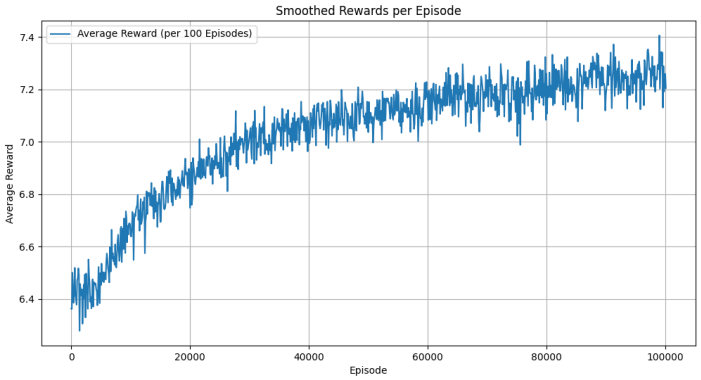

Figura 6.1: Andamento della funzione di reward nel tempo.

L'andamento delle metriche nel corso degli episodi è illustrando nella figura 6.2, che evidenzia la tendenza di tali indicatori a convergere verso i valori limite prestabiliti. Questo comportamento conferma il corretto apprendimento da parte dell'agente. Tuttavia, si osservano aree con valori particolarmente elevati e alcuni picchi verso il basso, sinonimo di instabilità della policy e, talvolta, di indecisione da parte dell'agente. Questi fenomeni potrebbero derivare da aggiornamenti della policy eccessivamente aggressivi o da una mancata convergenza stabile delle metriche. Ciò suggerisce la necessità di prolungare la durata del training e di ridurre il tasso di aggiornamento della policy, al fine di ottenere risultati più stabili e un comportamento più coerente da parte dell'agente.
Inoltre, potrebbe essere utile valutare se la complessità del task proposto al modello sia appropriata per il numero di episodi di addestramento considerati. Una complessità eccessiva, in combinazione con un numero limitato di step per episodio, potrebbe limitare la capacità dell'agente di esplorare soluzioni alternative, causando l'oscillazione osservata nei valori.

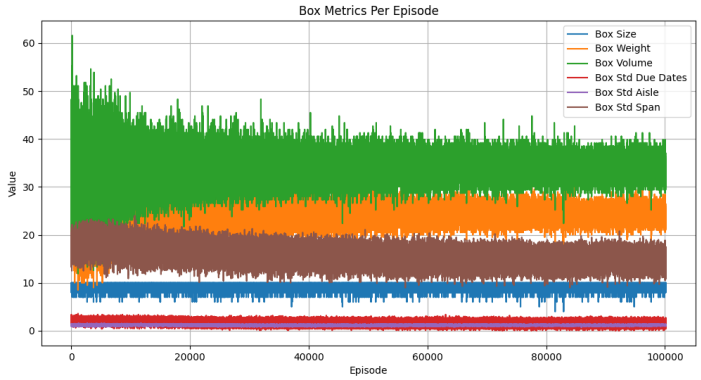

Figura 6.2: Andamento delle metriche nel tempo.

Il grafico seguente illustra l'andamento della perdita di valore nel tempo. La *value loss* rappresenta l'errore tra il valore previsto dalla funzione di valore e il valore effettivo, cioè

il valore calcolato sulla base delle ricompense ottenute dall'agente durante l'interazione con l'ambiente. L'evoluzione di tale fattore è quella desiderata, con un decremento esponenziale nella fase iniziale, per poi mantenersi su valori prossimi allo zero per la restante durata del training. Inoltre, l'assenza di picchi suggerisce buona stabilità della policy negli aggiornamenti.

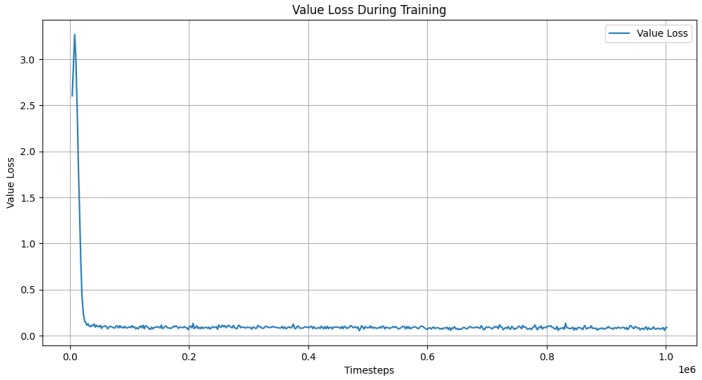

Figura 6.3: Andamento della value loss nel tempo.

La figura 6.4 rappresenta l'andamento della perdita di entropia nel corso degli episodi. L'*entropy loss* misura l'incertezza o la casualità nelle decisioni dell'agente. Un valore molto negativo incentiva l'agente a esplorare un'ampia gamma di azioni, mentre un valore prossimo allo zero indica che l'agente sta diventando troppo "deterministico", ovvero tende a scegliere sempre le stesse azioni. Dal grafico, risulta evidente la necessità di prolungare la durata del training, per consentire all'agente una fase di esplorazione più duratura, consentendogli di giungere a soluzioni migliori. Tuttavia, l'assenza di picchi e irregolarità suggeriscono una policy consistente a seguito di aggiornamenti.

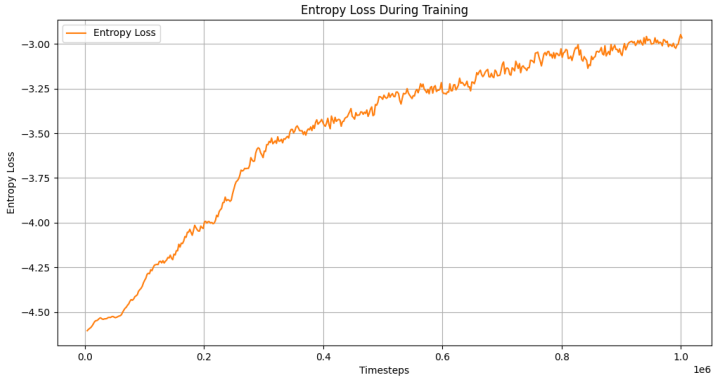

Figura 6.4: Andamento della perdita di entropia nel tempo.

I parametri rilevanti nella determinazione della stabilità della policy a seguito dell'apprendimento sono rappresentati nelle figure 6.5 e 6.6. La *policy loss* rappresenta l'errore

associato all'aggiornamento della politica durante il processo di apprendimento. L'obiettivo è mantenere questo valore pressochè costante, con minime variazioni, il che indica che gli aggiornamenti della policy non sono troppo bruschi o aggressivi. Il grafico di seguito riportato mostra una discreta stabilità, con valori che oscillano all'interno di un range di compreso tra -0.01 e -0.02. Questo suggerisce che, nonostante ci siano delle fluttuazioni, gli aggiornamenti della policy non sono eccessivamente drastici, il che favorisce un apprendimento regolare. Tuttavia, c'è ancora spazio per miglioramenti, specialmente nella riduzione di queste oscillazioni, per ottenere aggiornamenti più raffinati e una stabilità ancora maggiore.

Figura 6.5: Andamento della perdita di gradiente della policy nel tempo.

Un ulteriore parametro cruciale per la robustezza della policy è l'*approximate KL divergence*. Tale metrica misura quanto la distribuzione delle azioni scelte dall'agente differisce dalla precedente distribuzione dopo che la policy è stata aggiornata. L'Approximate KL Divergence permette di valutare quanto drasticamente la nuova policy si discosta dalla vecchia, e un valore stabile e basso è desiderabile per garantire aggiornamenti graduali e controllati. Il grafico in figura 6.6 mostra un'oscillazione più marcata rispetto alla policy loss, con valori che variano tra 0.004 e 0.012, e alcuni picchi che superano questa soglia. Queste fluttuazioni potrebbero indicare che gli aggiornamenti della policy sono talvolta troppo aggressivi, causando instabilità. Sarà quindi necessario intervenire sui parametri del modello, come il learning rate o il clip range, per ridurre queste oscillazioni e ottenere un comportamento più coerente e controllato nel corso del training. Un controllo più stretto sulla KL divergence è fondamentale per evitare che la policy subisca cambiamenti troppo rapidi, che potrebbero comprometterne la stabilità e l'efficienza a lungo termine.

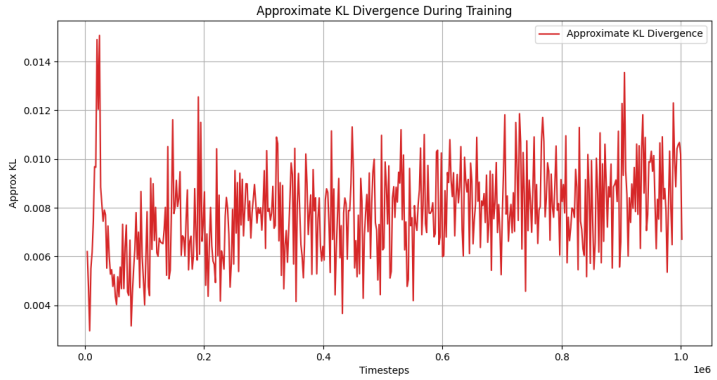

Figura 6.6: Andamento della divergenza KL nel tempo.

I risultati ottenuti a seguito di questo primo test hanno evidenziato la necessità di raffinare il valore attribuito ai parametri, ricercando maggiore stabilità negli aggiornamenti della policy. Ciò ha suggerito la possibilità di prolungare la durata del training, offrendo all'agente più tempo per esplorare soluzioni diverse, giungendo così a scelte più ottimali.

A seguito di numerosi test, si è giunti a risultati soddisfacenti, che mostrano un corretto apprendimento da parte del modelloe una stabilità significativamente migliorata. I parametri sono stati impostati con valori differenti, mantenendo gli stessi pesi:

- $max\_grad\_norm = 0.5$
- $batch\_size = 512$
- $learning\_rate = 1e - 5$
- $gamma = 0.99$
- $ent\_coef = 0.002$
- $gae\_lambda = 0.94$
- $clip\_range = 0.03$
- $n\_steps = 2048$
- $Step\_per\_episodio = 10$
- $Step = 4.500.000$

Il periodo di training è stato notevolmente prolungato, permettendo all'agente di esplorare una vasta gamma di strategie prima di convergere su una soluzione ottimale. I parametri sono stati progressivamente adattati, sulla base dei risultati dei test, con l'obiettivo di mantenere stabilità negli aggiornamenti della policy e garantire un apprendimento efficace da parte dell'algoritmo.

I grafici riportati di seguito evidenziano chiaramente tali miglioramenti.

I grafici nelle figure 6.7 e 6.8 mostrano i miglioramenti avvenuti in termini di stabilità e prestazioni. La reward function presenta una crescita molto più rapida rispetto ai test iniziali, raggiungendo valori superiori a 8 (su un massimo di 10). Parallelamente, le metriche chiave del modello convergono su un range più ristretto di valori, senza particolari picchi che indichino instabilità. Questo comportamento dimostra l'efficacia dell'ottimizzazione dei parametri nel garantire stabilità alla policy e migliorare il processo di apprendimento complessivo dell'agente.

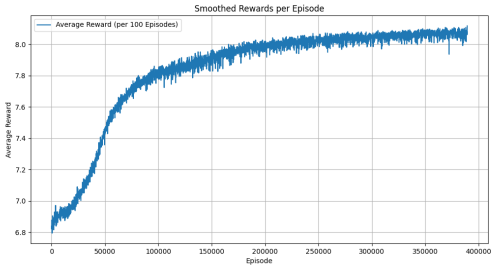

Figura 6.7: Andamento della funzione di reward nel tempo.

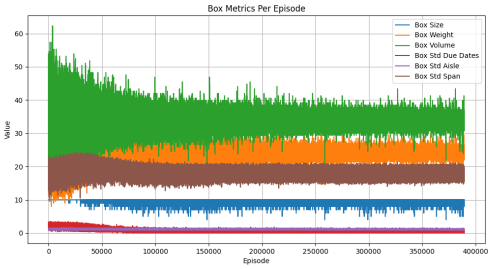

Figura 6.8: Andamento delle metriche nel tempo.

La perdita di valore nel tempo mostra un andamento simile a quello osservato precedentemente, ma con valori ancora più vicini allo zero, il che rappresenta un miglioramento significativo. Un valore di value loss vicino allo zero indica che la funzione di valore dell'agente è in grado di approssimare molto bene le ricompense future, segno che l'algoritmo sta imparando in modo efficace e stabile.

Per quanto riguarda l'entropy loss, dopo una fase iniziale di esplorazione significativa (visibile nella parte inferiore del grafico), l'andamento diventa debolmente crescente. Questo suggerisce che l'algoritmo continua a mantenere un livello di esplorazione, evitando di diventare eccessivamente deterministico troppo presto. La tendenza a un lieve aumento dell'entropia è indicativa del fatto che l'agente sta esplorando ulteriori opzioni, pur consolidando le azioni che hanno già portato a maggiori ricompense fino a quel momento. In generale, questo comportamento bilanciato tra esplorazione e sfruttamento è desiderabile, poiché consente all'agente di migliorare progressivamente le sue decisioni senza convergere prematuramente su strategie subottimali.

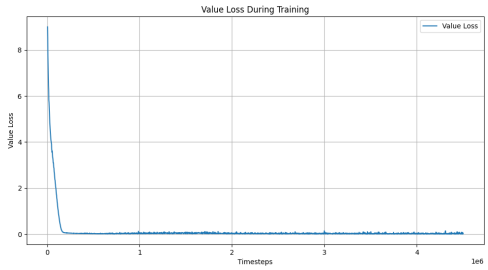

Figura 6.9: Andamento della value loss nel tempo.

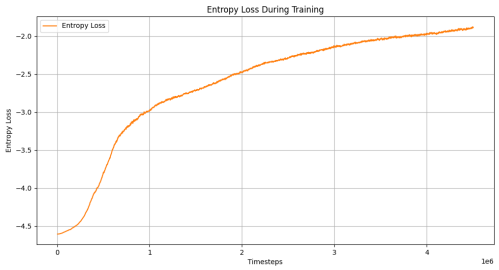

Figura 6.10: Andamento della perdita di entropia nel tempo.

I grafici relativi alla policy loss e alla divergenza KL approssimata presentano, a loro volta, un ulteriore perfezionamento. In particolare, la figura 6.11 illustra l'andamento della perdita di gradiente della policy nel tempo, con valori debolmente negativi e racchiusi in un intervallo estremamente ristretto (tra -0.0015 e -0.0025). Questo risultato conferma i miglioramenti osservati nei grafici precedenti, dimostrando una maggiore stabilità della policy a seguito di ogni aggiornamento.

Analogamente, il grafico della divergenza KL approssimata (figura 6.12) evidenzia un significativo restringimento del range di variabilità di tale parametro, che ora oscilla tra 0.00015 e 0.00030. Questo comportamento testimonia i miglioramenti apportati in termini di stabilità della policy, riducendo il grado di cambiamento tra la vecchia e la nuova policy, come desiderato.

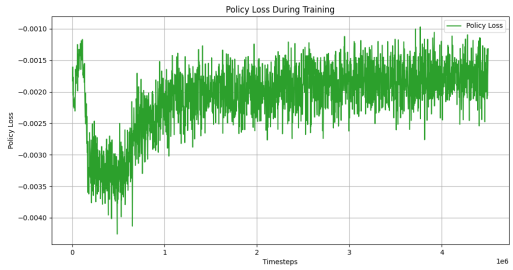

Figura 6.11: Andamento della perdita di gradiente della policy nel tempo.

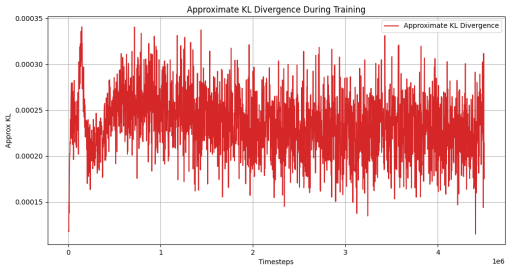

Figura 6.12: Andamento della divergenza KL nel tempo.

Conclusa la fase di fine-tuning dei parametri del modello, il passo successivo ha riguardato la valutazione delle prestazioni del sistema a seguito dell'integrazione dello Smart waving, rispetto al metodo tradizionale, che considera il solo Cluster Optimizer per la pianificazione dei task di picking. Questo confronto ha permesso di evidenziare i principali vantaggi introdotti dal nuovo modello, sia in termini di efficienza operativa che di riduzione dei tempi e delle distanze percorse dagli operatori.

6.1.2 Analisi delle prestazioni su 10 ordini

La valutazione delle prestazioni del modello è stata condotta tramite un confronto tra i risultati ottenuti a seguito dell'integrazione dello Smart waving nel processo e il metodo tradizionale. La metodologia attualmente usata per la pianificazione, come discusso nei capitoli precedenti, prevede l'utilizzo del Cluster Optimizerper generare i task di picking a partire dall'ondata di ordini ricevuta in input.

L'analisi è stata realizzata prendendo in considerazione le seguenti metriche chiave utilizzate nello sviluppo del modello di reinforcement learning:

- Numero di ordini presenti nel task n.
- Peso complessivo del cluster.
- Volume totale del raggruppamento.
- Deviazione standard della *due date* (σ_{DD}).
- Deviazione standard del numero di corridoi visitati (σ_A).
- Deviazione standard del numero di campate visitate (σ_S).

Tali indicatori sono stati quantificati sia per il task generato a seguito dell'introduzione dello Smart waving Il planning task generato con il metodo tradizionale è composto selezionando i primi n ordini della lista, in modo da replicare il comportamento di tale strumento nel suo tipico funzionamento.

Questa analisi iniziale ha previsto un n posto pari a 10, semplificando così il problema per ottenere un risultato preliminare su un numero limitato di ordini. Successivamente, sono stati condotti ulteriori test con un numero maggiore di picking order, per valutare la scalabilità dello strumento.

I risultati del confronto sono riportati nella tabella 6.1.

	Numero di ordini	**Peso (kg)**	**Volume (cm^3)**	σ_{DD} **(h)**	σ_A	σ_S
Caso ottimo	10	25	40	0	0	0
Caso con Smart Waving	10	25	35	0.3	1.2	19.3
Caso metodo tradizionale	10	33	46.2	2	1.2	16.4

Tabella 6.1: Confronto tra Smart Waving e metodo tradizionale.

I risultati riportati in tabella evidenziano i benefici derivanti dall'introduzione dello Smart waving nella pianificazione dei task di picking. Il peso del cluster raggiunge il valore ottimale di 25 kg, garantendo la massima efficienza nel riempimento del carrello destinato

al trasporto e il volume si avvicina al limite massimo (35 su 40 cm^3). Un ulteriore miglioramento significativo riguarda la riduzione della deviazione standard della due date, che risulta prossima allo zero, indicando una gestione più uniforme delle scadenze.

D'altro canto, il metodo tradizionale presenta diverse inefficienze. In particolare, eccede i limiti prestabiliti di peso e volume, il che comporta la creazione di due task separati a partire dalla stessa lista di ordini. Questo porta a un utilizzo maggiore delle risorse e impedisce un'efficace ottimizzazione del riempimento dei mezzi di movimentazione.

Inoltre, è stata condotta un'ulteriore analisi per approfondire i vantaggi offerti dallo Smart Waving nel contesto della pianificazione dei task di picking. Lo studio svolto è stato strutturato in modo analogo al precedente, ma considerando dei limiti inferiori, rispetto al primo caso analizzato, per le variabili rilevanti. L'obiettivo di tale attività è stato quello di valutare la flessibilità del modello e la sua capacità di mantenere prestazioni soddisfacenti anche con vincoli più stringenti. I risultati dell'analisi sono riportati nella tabella 6.2.

	Numero di ordini	**Peso (kg)**	**Volume (cm^3)**	σ_{DD} **(h)**	σ_A	σ_S
Caso ottimo	10	20	35	0	0	0
Caso con Smart Waving	10	20	28	0.4	1	20
Caso metodo tradizionale	10	33	46.2	2	1.2	16.4

Tabella 6.2: Confronto tra Smart Waving e metodo tradizionale con limiti più stringenti.

I risultati evidenziati in tabella confermano ulteriormente le superiori prestazioni dello Smart waving rispetto al metodo tradizionale. La flessibilità del modello ha permesso la realizzazione di un raggruppamento di ordini in grado di adattarsi efficacemente al cambiamento delle esigenze aziendali, rappresentate dalla modifica dei vincoli di peso e volume. Al contrario, Il processo attuale ha riproposto il medesimo cluster, con inefficienze più marcate che comportano la creazione di ulteriori task di picking e, di conseguenza, un maggiore impiego di risorse. In particolare, la capacità del modello di ridurre la deviazione standard della data di scadenza degli ordini dimostra come l'algoritmo favorisca l'aggregazione di ordini con scadenze simili, migliorando così la coerenza temporale delle operazioni. Questo porta a un miglior utilizzo delle risorse di magazzino e una riduzione dei tempi di attesa nelle fasi di spedizione. Inoltre, il rispetto dei limiti di peso e volume previene l'inefficienza legata a task sovraccaricati o incompleti, ottimizzando il flusso di lavoro.

Le prestazione del modello si sono dimostrate estreamente soddisfacenti, portando a un miglioramento significative nella qualità dei cluster generati. Questo, in termini di ri-

spetto dei vincoli stabiliti, si traduce in numerosi benefici operativi per l'azienda che implementa lo Smart waving.

6.2 Benefici e limitazioni del modello

Attualmente, il processo di generazione dei task di picking presenta diverse inefficienze, in gran parte dovute alla metodologia utilizzata per la pianificazione dei picking order, che andranno poi a comporre i task. Sebbene il Cluster Optimizer abbia avuto un impatto positivo sull'efficienza, ottimizzando l'assegnazione degli ordini ai rispettivi cluster, il metodo di pianificazione è rimasto invariato, lasciando così spazio a ulteriori miglioramenti.

Lo Smart Waving si propone di risolvere le problematiche della pianificazione tradizionale dei task di picking. Questo strumento individua, a partire da una lista di ordini, i candidati ideali per formare un task ottimale, definito in base ai valori delle metriche rilevanti nel processo: dimensione, peso e volume del cluster, deviazione standard della data di scadenza (*due date*), del numero di corridoi e del numero di campate. Questa modalità operativa permette di superare la selezione casuale tipica del metodo tradizionale, effettuando scelte orientate all'ottimizzazione complessiva del processo.

I risultati ottenuti evidenziano chiaramente i benefici apportati dallo Smart Waving al picking. Tra i vantaggi si annoverano il miglior riempimento dei mezzi di movimentazione utilizzati per il trasporto, la riduzione del numero di task da eseguire e, di conseguenza, un minor impiego di risorse, con un conseguente risparmio nei costi di gestione e un generale miglioramento dell'efficienza operativa del magazzino.

Tuttavia, essendo ancora in una fase preliminare, questo strumento presenta diverse limitazioni che possono scoraggiarne l'implementazione. Il modello, infatti, è complesso da sviluppare e richiede competenze specialistiche in algoritmi di machine learning, rendendolo meno accessibile per un'ampia gamma di aziende. Inoltre, l'analisi è stata condotta con alcune semplificazioni, come un numero ridotto di ordini. Nei magazzini di grandi aziende, dove gli ordini possono superare il migliaio, il tempo di calcolo per il training dell'algoritmo cresce esponenzialmente.

La principale criticità dello Smart Waving risiede nella sua scarsa scalabilità, dovuta alla struttura stessa dell'algoritmo. Il modello è stato progettato affinché l'agente selezioni le azioni scegliendo il codice identificativo di un ordine. Di conseguenza, durante gli episodi successivi, l'agente tende a selezionare l'ID degli ordini che generano le maggiori ricompense. Tuttavia, questa modalità di operare implica che, cambiando la lista iniziale degli ordini, sarà necessario ripetere completamente il training, limitando fortemente

l'applicabilità del modello in scenari reali e dinamici. A supporto di queste osservazioni, verrà mostrato di seguito un caso di studio.

6.2.1 Analisi delle prestazione con più di 10 ordini

Nel corso dell'analisi delle prestazioni del modello, allo scopo di valutarne la scalabilità, sono stati eseguiti alcuni test con un numero di ordini superiore a 10. In particolare, n è staqto posto pari a 30. I risultati ottenuti evidenziano i limiti di questa prima versione dello Smart waving: con l'aumento del numero di ordini da gestire, la stabilità della policy del modello tende a peggiorare durante la fase di training.
I parametri utilizzati per il test sono stati impostati come segue:

- $max\_grad\_norm = 0.5$
- $batch\_size = 512$
- $learning\_rate = 2e - 5$
- $gamma = 0.99$
- $ent\_coef = 0.003$
- $gae\_lambda = 0.94$
- $clip\_range = 0.02$
- $n\_steps = 2048$
- $Step\_per\_episodio = 35$
- $Step = 70.000.000$

I pesi assegnati alle variabili rilevanti nel processo sono stati mantenuti pressochè invariati, con alcune leggere variazioni in alcuni test, per valutarne l'effetto sulla modalità di apprendimento dell'agente:

- Dimensione del cluster: 4.
- Peso del cluster: 3.
- Volume del cluster: 3.
- Deviazione standard della due date all'interno del ragruppamento: 4.
- Deviazione standard del numero di corridoi all'interno del ragruppamento: 1
- Deviazione standard del numero di campate all'interno del ragruppamento: 1

Di seguito sono riportati i grafici ottenuti durante la fase di test. Il grafico della funzione di reward mostra un andamento crescente, come previsto, a dimostrazione del corretto ap-

prendimento dell'agente. Tuttavia, durante la fase di training si osservano diversi picchi, nonostante la prolungata fase di fine-tuning con varie combinazioni di parametri. Questi picchi suggeriscono una certa instabilità negli aggiornamenti della policy, in particolare durante i periodi di esplorazione.

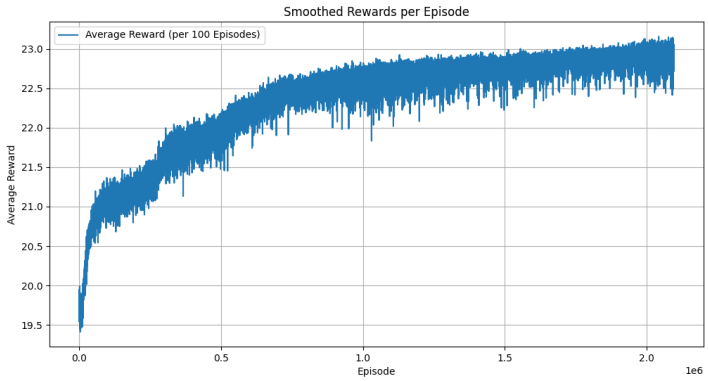

Figura 6.13: Andamento della funzione di reward nel tempo.

Il grafico seguente illustra l'andamento della perdita di valore (*value loss*) nel tempo. Tale andamento è caratterizzato da una notevole instabilità, con numerosi picchi che si presentano durante l'intero training, confermando la difficoltà del modello nel gestire un numero più elevato di ordini.

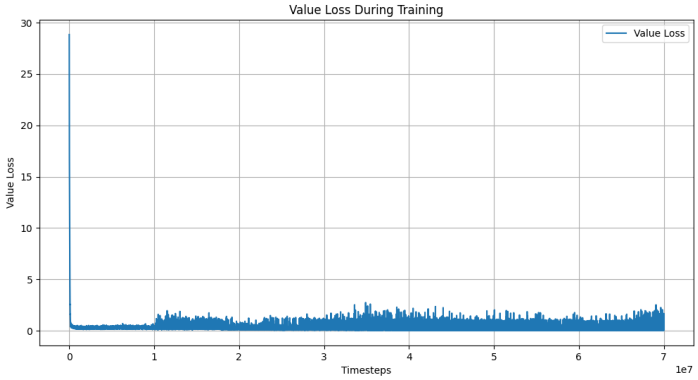

Figura 6.14: Andamento della value loss nel tempo.

Il grafico della *entropy loss* (Figura 6.15) mostra un andamento conforme alle aspettative: una crescita iniziale, che incentiva l'esplorazione dell'agente, seguita da una stabilizzazione su un certo valore. Tuttavia, anche qui si notano picchi e irregolarità, suggerendo che la policy non è ancora completamente stabile a seguito degli aggiornamenti.

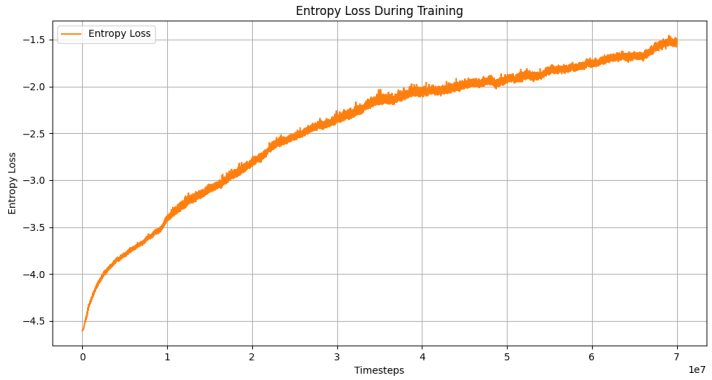

Figura 6.15: Andamento della perdita di entropia nel tempo.

I grafici della *policy loss* e della *approximate KL divergence* evidenziano ulteriormente l'instabilità del modello durante la fase di training. Le fluttuazioni significative e i picchi in entrambi i grafici sottolineano come l'algoritmo richieda un ulteriore fine-tuning per raggiungere prestazioni soddisfacenti, specialmente in scenari con un numero di ordini crescente.

Figura 6.16: Andamento della perdita di gradiente della policy nel tempo.

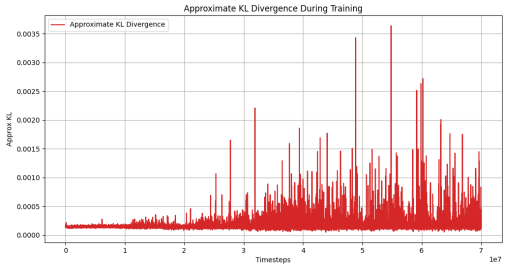

Figura 6.17: Andamento della divergenza KL nel tempo.

I risultati di questi test confermano che, sebbene lo Smart waving apporti benefici significativi in scenari con un numero ridotto di ordini, l'aumento degli ordini evidenzia alcune limitazioni strutturali nel modello attuale. L'instabilità della policy, come risulta dagli andamenti irregolari della reward e delle altre metriche, suggerisce la necessità di continuare il fine-tuning dei parametri. Tuttavia, l'attuale versione del modello mostra difficoltà nell'adattarsi a un ambiente complesso e dinamico, come evidenziato dai picchi nella policy loss e nella divergenza KL.

Questi risultati sottolineano che, per estendere lo Smart waving a scenari reali e su larga scala, è necessario introdurre miglioramenti che aumentino la scalabilità e la flessibilità del modello, che verranno discussi nella sezione successiva.

6.3 Sviluppi futuri

Le criticità emerse evidenziano i limiti di questo modello, suggerendo la necessità di sviluppare una versione più avanzata e versatile, in grado di superare tali restrizioni. L'elemento principale che ostacola l'implementazione di questo algoritmo su larga scala è la sua scarsa flessibilità e scalabilità. L'impossibilità di adattare facilmente lo Smart waving a diversi contesti aziendali limita notevolmente il suo potenziale utilizzo e impedisce di sfruttare a pieno i benefici della pianificazione ottimizzata.

L'obiettivo degli studi futuri nella pianificazione dei task di picking è quindi di affrontare e risolvere questi problemi, sviluppando un modello commercializzabile, in grado di adattarsi a vari scenari senza richiedere un lungo processo di training. Attualmente, l'azienda sta focalizzando la sua ricerca su uno strumento capace di identificare autonomamente gli ordini più adatti da includere in un task, basandosi sulle caratteristiche specifiche di ciascun ordine. Questa modifica apporterebbe un miglioramento significativo, rendendo lo Smart Waving applicabile in contesti aziendali differenti e ottimizzando ulteriormente il processo di pianificazione.

L'introduzione di questo tool consentirebbe all'azienda di commercializzare un prodotto con numerosi benefici, estremamente adattabile e con l'unica criticità legata alla complessità computazionale e allo sviluppo di un modello di reinforcement learning avanzato. Il funzionamento della versiona aggiornata dello Smart waving si basa su un processo di addestramento dell'agente lungo e complesso, volto a fargli apprendere quali ordini selezionare in base ai parametri di peso, volume e data di spedizione che li caratterizzano. Per implementare queste funzionalità, l'agente deve analizzare questi indicatori per ciascun ordine, sperimentando diverse combinazioni nella creazione del cluster ottimale.

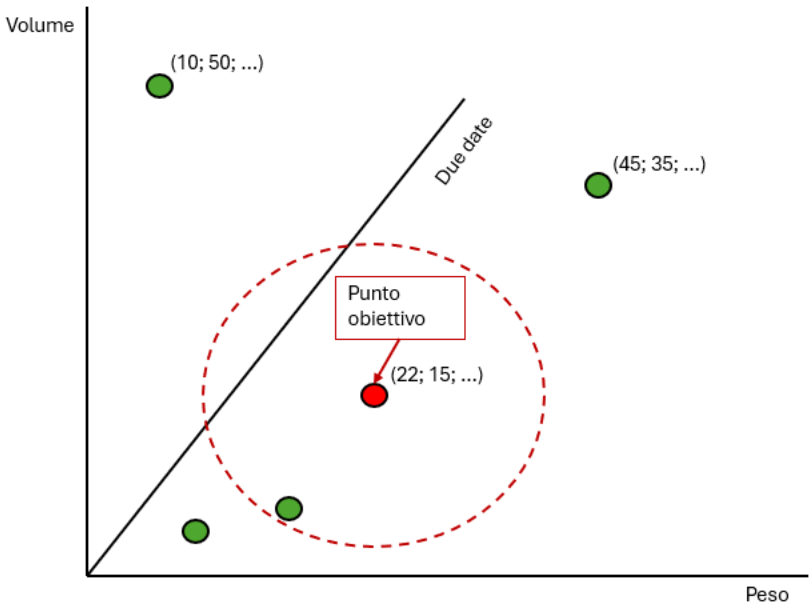

Figura 6.18: Rappresentazione del funzionamento dello versione 2.0 del tool.

La figura 6.18 illustra il funzionamento dello Smart waving 2.0. Inizialmente, viene stabilito un obiettivo in termini di valori ottimali per le metriche in esame, rappresentato dal punto rosso nell'immagine. Successivamente l'algoritmo procede a selezionare gli ordini che si trovano in un intorno accettabile rispetto a questi valori, creando così il cluster ottimale con gli ordini a disposizione.

Lo strumento presentato in questo documento mira a ridurre le inefficienze legate al processo di pianificazione dei task di picking, con risultati estremamente soddisfacenti. Tuttavia, esistono diverse limitazioni che rappresentano un ostacolo all'implementazione di questo modello all'interno di un reale contesto aziendale. Allo scopo di superare tali criticità, l'azienda ha intrapreso un percorso di ricerca volto allo sviluppo di una versione avanzata dello Smart Waving, dotata di una maggiore flessibilità e in grado di adattarsi a differenti scenari senza la necessità di lunghi processi di addestramento. Questo upgrade consentirebbe di trasformare il modello in un prodotto commercializzabile e altamente adattabile, rispondendo così alle esigenze di un mercato dinamico e rendendo il tool un elemento di grande valore per l'ottimizzazione logistica.

Conclusioni

La presente tesi ha analizzato le fasi che hanno condotto alla realizzazione dello Smart waving, mostrando come l'utilizzo del reinforcement learning all'interno della catena logistica di un'azienda, in particolare nella gestione del magazzino, rappresenti la "next big thing" nel mondo della logistica, con significativi benefici in termini di efficienza, flessibilità e adattabilità del processo di picking. A seguito di un'introduzione dei concetti fondamentali di intelligenza artificiale e le sue applicazioni nel contesto logistico, sono state prese in considerazione le esigenze operative specifiche che hanno condotto alla progettazione del modello.

La definizione della struttura dell'algoritmo è stata svolta seguendo un percorso complesso e articolato, che ha richiesto un'analisi preliminare di fattibilità prima di poter procedere alla definizione delle componenti costitutive del modello - reward function, action space, observation space. La fase successiva è stata l'implementazione del codice, agevolata da un approccio modulare e ispirata dalle consolidate librerie Gym di OpenAI, che fornisce un ambiente flessibile, scalabile, personalizzabile alle necessità specifiche e adatto al training di questa famiglia di algoritmi.

Il testing del modello ha evidenziato le problematiche riguardanti i valori attribuiti ai parametri tecnici del modello, sottolineando la necessità di realizzare un processo di fine-tuning, allo scopo di ottenere una policy stabile e performante. Durante la fase di training, è stata posta l'attenzione sulla combinazione di valori assunti da tali indicatori, ricercando l'equilibrio ottimale che garantisse stabilità e un corretto processo di apprendimento da parte dell'agente. L'ottimizzazione dei parametri ha portato a un graduale miglioramento nella modalità di operare da parte del modello, validando l'approccio considerato.

I vantaggi derivanti dall'introduzione dello Smart waving nel processo di pianificazione sono significativi e includono un maggiore riempimento dei mezzi destinati al trasporto, una riduzione del numero di task di picking, un migliore utilizzo delle risorse a disposizione e un efficientamento generale del processo. I test dimostrano che l'uso di questo strumento è adatto a un contesto con un ridotto numero di ordini da prelevare, evidenziando i limiti nella scalabilità e nella stabilità di tale modello quando il volume di picking order è elevato.

Nonostante i benefici evidenziati, il modello presenta delle criticità che ne ostacolano la diffusione in contesti aziendali reali, dalla necessità di una prolungata fase di training alla difficoltà ad adattarsi a scenari dinamici e in continuo cambiamento, fino alla complessità computazionale che caratterizza questa tipologia di algoritmi. La mancanza di flessibilità e scalabilità, in particolare, rappresenta un elemento cruciale per magazzini che necessitano un sistema per la gestione un volume di ordini elevato.

L'obiettivo futuro è il superamento di queste limitazioni, realizzando un modello in grado di essere applicato a vari scenari con problematiche di gestione più o meno complesse. La ricerca, dunque, si sta orientando verso l'elaborazione di un algoritmo più versatile, che non richieda un lungo processo di addestramento precedente alla sua implementazione. In quest'ottica, la modularizzazione del modello, con l'introduzione di componenti separati per la selezione degli ordini e la gestione dei vincoli di peso e volume, si configura come una soluzione promettente.

Concludendo, lo Smart waving fornisce una proposta interessante per l'automazione e l'ottimizzazione del processo di picking, un ambito che da sempre rappresenta un fattore critico nella ricerca dell'efficienza. Nonostante i miglioramenti necessari per renderlo uno strumento commercializzabile, i risultati sono estremamente soddisfacenti e sottolineano le potenzialità del reinforcement learning nel processo di *digital transformation* di una supply chain. L'obiettivo ultimo di questo progetto è la creazione di un sistema adattabile e scalabile, in grado di rispondere alle mutevoli esigenze del mercato e che generi un vantaggio competitivo per le aziende che decideranno di introdurlo all'interno dei loro processi.

Bibliografia

[1] B. Buchmeister, I. Palcic, and R. Ojstersek, "Artificial intelligence in manufacturing companies and broader: An overview," *DAAAM International Scientific Book*, pp. 81–98, 2019.

[2] Gartner, "Gartner identifies top trends in supply chain technology for 2024," 2024. Accessed: 2024-09-26.

[3] Z.-H. Zhou, *Machine learning*. Springer nature, 2021.

[4] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.

[5] E. B. Tirkolaee, S. Sadeghi, F. M. Mooseloo, H. R. Vandchali, and S. Aeini, "Application of machine learning in supply chain management: a comprehensive overview of the main areas," *Mathematical problems in engineering*, vol. 2021, no. 1, p. 1476043, 2021.

[6] S. Makkar, G. N. R. Devi, and V. K. Solanki, "Applications of machine learning techniques in supply chain optimization," in *ICICCT 2019–System Reliability, Quality Control, Safety, Maintenance and Management: Applications to Electrical, Electronics and Computer Science and Engineering*, pp. 861–869, Springer, 2020.

[7] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] I. Goodfellow, "Deep learning," 2016.

[10] B. P. Bhattarai, S. Paudyal, Y. Luo, M. Mohanpurkar, K. Cheung, R. Tonkoski, R. Hovsapian, K. S. Myers, R. Zhang, P. Zhao, *et al.*, "Big data analytics in smart grids: state-of-the-art, challenges, opportunities, and future directions," *IET Smart Grid*, vol. 2, no. 2, pp. 141–154, 2019.

[11] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[13] D. Ernst and A. Louette, "Introduction to reinforcement learning," *Feuerriegel, S., Hartmann, J., Janiesch, C., and Zschech, P*, pp. 111–126, 2024.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[16] R. Carbonneau, K. Laframboise, and R. Vahidov, "Application of machine learning techniques for supply chain demand forecasting," *European journal of operational research*, vol. 184, no. 3, pp. 1140–1154, 2008.

[17] S. Chopra and P. Meindl, *Supply chain management. Strategy, planning & operation.* Springer, 2007.

[18] Y. S. Lee and R. Sikora, "Application of adaptive strategy for supply chain agent," *Information Systems and e-Business Management*, vol. 17, pp. 117–157, 2019.

[19] T. Abu Zwaida, C. Pham, and Y. Beauregard, "Optimization of inventory management to prevent drug shortages in the hospital supply chain," *Applied Sciences*, vol. 11, no. 6, p. 2726, 2021.

[20] M. T. Afridi, S. Nieto-Isaza, H. Ehm, T. Ponsignon, and A. Hamed, "A deep reinforcement learning approach for optimal replenishment policy in a vendor managed inventory setting for semiconductors," in *2020 Winter Simulation Conference (WSC)*, pp. 1753–1764, IEEE, 2020.

[21] B. Rolf, I. Jackson, M. Müller, S. Lang, T. Reggelin, and D. Ivanov, "A review on reinforcement learning algorithms and applications in supply chain management," *International Journal of Production Research*, vol. 61, no. 20, pp. 7151–7179, 2023.

[22] E. Gutierrez-Franco, C. Mejia-Argueta, and L. Rabelo, "Data-driven methodology to support long-lasting logistics and decision making for urban last-mile operations," *Sustainability*, vol. 13, no. 11, p. 6230, 2021.

[23] A. Bretas, A. Mendes, S. Chalup, M. Jackson, R. Clement, and C. Sanhueza, "Modelling railway traffic management through multi-agent systems and reinforcement learning," in *In Elsawah, S.(ed.) MODSIM2019, 23rd International Congress on Modelling and Simulation*, pp. 291–297, Modelling and Simulation Society of Australia and New Zealand, Canberra ..., 2019.

[24] E. Puskás, Á. Budai, and G. Bohács, "Optimization of a physical internet based supply chain using reinforcement learning," *European Transport Research Review*, vol. 12, no. 1, p. 47, 2020.

[25] C.-F. Chien, Y.-S. Lin, and S.-K. Lin, "Deep reinforcement learning for selecting demand forecast models to empower industry 3.5 and an empirical study for a semiconductor component distributor," *International Journal of Production Research*, vol. 58, no. 9, pp. 2784–2804, 2020.

[26] W. Dangelmaier, T. Rust, A. Doring, and B. Klopper, "A reinforcement learning approach for learning coordination rules in production networks," in *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*, pp. 84–84, IEEE, 2006.

[27] C. De Maio, G. Fenza, V. Loia, F. Orciuoli, and E. Herrera-Viedma, "A framework for context-aware heterogeneous group decision making in business processes," *Knowledge-Based Systems*, vol. 102, pp. 39–50, 2016.

[28] T. Kim, R. U. Bilsel, and S. Kumara, "Supplier selection in dynamic competitive environments," *International Journal of Services Operations and Informatics*, vol. 3, no. 3-4, pp. 283–293, 2008.

[29] H. Du and Y. Jiang, "Backup or reliability improvement strategy for a manufacturer facing heterogeneous consumers in a dynamic supply chain," *IEEE Access*, vol. 7, pp. 50419–50430, 2019.

[30] C. Liu, "Outsourcing strategies for manufacturers facing reputation oriented consumers," in *2020 Chinese Automation Congress (CAC)*, pp. 1980–1985, IEEE, 2020.

[31] J. Reeder, G. Sukthankar, M. Georgiopoulos, and G. Anagnostopoulos, "Intelligent trading agents for massively multi-player game economies," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, pp. 102–107, 2008.

[32] K. C. Chatzidimitriou, A. L. Symeonidis, and P. A. Mitkas, "Policy search through adaptive function approximation for bidding in tac scm," in *International Workshop on Agent-Mediated Electronic Commerce*, pp. 16–29, Springer, 2012.

[33] H. Du and T. Xiao, "Pricing strategies for competing adaptive retailers facing complex consumer behavior: Agent-based model," *International Journal of Information Technology & Decision Making*, vol. 18, no. 06, pp. 1909–1939, 2019.

[34] H. Liu, E. Howley, and J. Duggan, "An agent-based simulation of the effects of consumer behavior on market price dynamics," in *Proceedings of the IASTED International Conference on Applied Simulation and Modelling, ASM 2011, Crete, Greece*, pp. 316–325, 2011.

[35] D. M. Lambert, "Fundamentals of logistics management," 1998.

[36] J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco, *Facilities planning*. John Wiley & Sons, 2010.

[37] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse operation: A comprehensive review," *European journal of operational research*, vol. 177, no. 1, pp. 1–21, 2007.

[38] L. Y. Tsui and C.-H. Chang, "A microcomputer based decision support tool for assigning dock doors in freight yards," *Computers & Industrial Engineering*, vol. 19, no. 1-4, pp. 309–312, 1990.

[39] K. R. Gue, "The effects of trailer scheduling on the layout of freight terminals," *Transportation Science*, vol. 33, no. 4, pp. 419–428, 1999.

[40] J. J. Bartholdi III and K. R. Gue, "Reducing labor costs in an ltl crossdocking terminal," *Operations research*, vol. 48, no. 6, pp. 823–832, 2000.

[41] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European journal of operational research*, vol. 182, no. 2, pp. 481–501, 2007.

[42] A. Ramaa, K. Subramanya, and T. Rangaswamy, "Impact of warehouse management system in a supply chain," *International Journal of Computer Applications*, vol. 54, no. 1, 2012.

[43] C. G. Petersen, G. R. Aase, and D. R. Heiser, "Improving order-picking performance through the implementation of class-based storage," *International Journal of Physical Distribution & Logistics Management*, vol. 34, no. 7, pp. 534–544, 2004.

[44] M. J. Rosenblatt and Y. Roll, "Warehouse design with storage policy considerations," *The International Journal of Production Research*, vol. 22, no. 5, pp. 809–821, 1984.

[45] P. J. Parikh and R. D. Meller, "Selecting between batch and zone order picking strategies in a distribution center," *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 5, pp. 696–719, 2008.

[46] R. A. Ruben and F. R. Jacobs, "Batch construction heuristics and storage assignment strategies for walk/ride and pick systems," *Management Science*, vol. 45, no. 4, pp. 575–596, 1999.

[47] E. Elsayed and M.-K. Lee, "Order processing in automated storage/retrieval systems with due dates," *IIE transactions*, vol. 28, no. 7, pp. 567–577, 1996.

[48] G. Cormier, "On the scheduling of order-picking operations in single-aisle automated storage and retrieval systems," *Modern Production Management Systems*, pp. 75–87, 1987.

[49] N. Gademann and S. Velde, "Order batching to minimize total travel time in a parallel-aisle warehouse," *IIE transactions*, vol. 37, no. 1, pp. 63–75, 2005.

[50] C. Theys, O. Bräysy, W. Dullaert, and B. Raa, "Using a tsp heuristic for routing order pickers in warehouses," *European Journal of Operational Research*, vol. 200, no. 3, pp. 755–763, 2010.

[51] G. Dukic and C. Oluic, "Order-picking methods: improving order-picking efficiency," *International Journal of Logistics Systems and Management*, vol. 3, no. 4, pp. 451–460, 2007.

[52] H. Hwang*, Y. Oh, and Y. Lee, "An evaluation of routing policies for order-picking operations in low-level picker-to-part system," *International journal of production research*, vol. 42, no. 18, pp. 3873–3889, 2004.

[53] C. G. Petersen, "An evaluation of order picking routeing policies," *International Journal of Operations & Production Management*, vol. 17, no. 11, pp. 1098–1111, 1997.

[54] G. Djukić and C. Oluić, "Heuristics, advanced heuristics or optimal algorithm," *Strojniški vestnik*, vol. 50, no. 11, pp. 530–535, 2004.

[55] F. Torchio, *Survey on automated systems for smart warehouses.* PhD thesis, Politecnico di Torino, 2023.

[56] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Artificial intelligence applications for industry 4.0: A literature-based study," *Journal of Industrial Integration and Management*, vol. 7, no. 01, pp. 83–111, 2022.

[57] B. Cheng, L. Wang, Q. Tan, and M. Zhou, "A deep reinforcement learning hyper-heuristic to solve order batching problem with mobile robots," *Applied Intelligence*, pp. 1–23, 2024.

[58] D. Silitonga, S. A. A. Rohmayanti, Z. Aripin, D. Kuswandi, A. B. Sulistyo, *et al.*, "Edge computing in e-commerce business: economic impacts and advantages of scalable information systems," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 11, no. 1, 2024.

[59] M. Akbari, "Revolutionizing supply chain and circular economy with edge computing: Systematic review, research themes and future directions," *Management Decision*, vol. 62, no. 9, pp. 2875–2899, 2024.

[60] M. Panzer and N. Gronau, "Designing an adaptive and deep learning based control framework for modular production systems," *Journal of Intelligent Manufacturing*, pp. 1–24, 2023.

[61] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[62] Y. Jiang, S. Yin, K. Li, H. Luo, and O. Kaynak, "Industrial applications of digital twins," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2207, p. 20200360, 2021.

[63] Z. Huang, Y. Shen, J. Li, M. Fey, and C. Brecher, "A survey on ai-driven digital twins in industry 4.0: Smart manufacturing and advanced robotics," *Sensors*, vol. 21, no. 19, p. 6340, 2021.

[64] J. Wan, X. Li, H.-N. Dai, A. Kusiak, M. Martinez-Garcia, and D. Li, "Artificial-intelligence-driven customized manufacturing factory: key technologies, applications, and challenges," *Proceedings of the IEEE*, vol. 109, no. 4, pp. 377–398, 2020.

[65] O. T. Gabriel, "Data privacy and ethical issues in collecting health care data using artificial intelligence among health workers," Master's thesis, Center for Bioethics and Research, 2023.

[66] Reply, "Company profile," 2024. Accessed: 2024-10-09.

[67] Gartner, "Gartner magic quadrant for warehouse management systems," 2024. Accessed: 2024-10-09.

[68] L. Reply, "Company profile," 2024. Accessed: 2024-10-09.

[69] G. Dunn, H. Charkhgard, A. Eshragh, S. Mahmoudinazlou, and E. Stojanovski, "Deep reinforcement learning for picker routing problem in warehousing," *arXiv preprint arXiv:2402.03525*, 2024.

[70] L. Begnardi, H. Baier, W. van Jaarsveld, and Y. Zhang, "Deep reinforcement learning for two-sided online bipartite matching in collaborative order picking," in *Asian Conference on Machine Learning*, pp. 121–136, PMLR, 2024.

[71] F. Niu and Z. Wang, "Model based reinforcement learning for simplified storage assignment optimization," 2019.

[72] E. Alkafaween, A. Hassanat, E. Essa, and S. Elmougy, "An efficiency boost for genetic algorithms: Initializing the ga with the iterative approximate method for optimizing the traveling salesman problem—experimental insights," *Applied Sciences*, vol. 14, no. 8, p. 3151, 2024.

[73] T. Guilmeau, E. Chouzenoux, and V. Elvira, "Simulated annealing: A review and a new scheme," in *2021 IEEE statistical signal processing workshop (SSP)*, pp. 101–105, IEEE, 2021.

[74] A. R. Khan, "Dynamic load balancing in cloud computing: Optimized rl-based clustering with multi-objective optimized task scheduling," *Processes*, vol. 12, no. 3, p. 519, 2024.

[75] Y. Saleem, K.-L. A. Yau, H. Mohamad, N. Ramli, M. H. Rehmani, and Q. Ni, "Clustering and reinforcement-learning-based routing for cognitive radio networks," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 146–151, 2017.

[76] OpenAI, "An api standard for reinforcement learning with a diverse collection of reference environments," 2024. Accessed: 2024-16-10.

[77] AWS, "Aws robomaker - esegui, scala e automatizza le simulazioni di robotica," 2024. Accessed: 2024-16-10.

[78] NVIDIA, "Nvidia isaac sim," 2024. Accessed: 2024-16-10.

[79] OpenAI, "Basic usage," 2024. Accessed: 2024-17-10.