Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea magistrale in Informatica

Relazione di Sistemi Complessi: Modelli e Simulazioni

Relazione di:
Preziosa Alessandro 866142
Refolli Francesco 865955

Anno Accademico 2023-2024

Indice

1 Introduzione

La **Swarm Robotics** è un filone di ricerca della robotica che si occupa di costruire sistemi intelligenti formati da una moltitudine di droni/robot indipendenti eventualmente eterogenei (detto sciame) per raggiungere tramite algoritmi fortemente decentralizzati un obiettivo di interesse collettivo. Le applicazioni della Swarm Robotics spaziano in campo civile e militare. Solitamente la ricerca si concentra su scenari che prevedono il salvataggio di persone (Search and Rescure), la ricognizione di un ambiente, o l'impiego operativo in battaglia. Gli algoritmi di Swarm Robotics possono anche avere ispirazione biologica[5].

Per la realizzazione di questi esperimenti sono stati costruiti diversi simulatori e qualche framework fisico per l'implementazione di sciami omogenei di robot.

L'obiettivo del progetto è simulare uno sciame di 20 droni volanti il cui scopo è:

1. dividersi in 5 squadroni (partizioni dell'insieme di droni)

2. decollare

3. raggiungere i 5 obiettivi sparsi nell'ambiente, evitando collisioni con altri droni durante il volo.

La collocazione degli obiettivi è stata generata casualmente usando una distribuzione uniforme sulle 3 componenti spaziali, i cui limiti sono gli stessi dei droni e dell'arena. Una partizione è considerata *buona* se la distribuzione dei droni sugli obiettivi è uniforme, ovvero se la varianza del numero di droni per squadrone è bassa.

Per la simulazione ci siamo avvalsi di ARGoS [4], un simulatore programmabile Free and Open Source.

2 ARGoS

ARGoS è un simulatore di droni che permette di utilizzare diversi engine fisici a 2 o 3 dimensioni e aggiungere programmaticamente plugin per nuovi droni o nuovi contenuti.

In figura 1 è riportato uno schema di funzionamento dell'engine di ARGoS da cui si possono trarre alcune osservazioni. Per comandare un drone si deve implementare un **Controller**, il quale tramite **Attuatori** e **Sensori** può interagine con l'ambiente e con gli altri droni. Solitamente questi componenti si appoggiano su un **Media** che costituisce il canale di comunicazione e storage delle informazioni. Un altro valido strumento fornito al programmatore sono le **Loop Functions**, funzioni che vengono eseguite ad ogni tick, all'inizio o alla fine della simulazione per consentire interazioni custom con l'ambiente o di inizializzare la simulazione (per esempio istanziando oggetti o facendo controlli). Una variante di queste sono le **User Functions**, che vengono solitamente impiegate per costruire effetti visivi in quanto permettono di interagire con lo stack grafico di ARGoS, per esempio disegnando label sulle entità. Si possono inoltre costruire delle **Loop Function**, funzioni solitamente utilizzate per interagire con la simulazione, e **User Functions**, funzioni che permettono di interagire con lo stack grafico di ARGoS. Queste integrazioni vengono realizzate sotto forma di plugins che possono essere caricati dinamicamente tramite una configurazione XML di una simulazione, che permette anche di disporre automaticamente delle entità nello spazio, di configurare i controller e abilitare le functions (non si possono utilizzare più di una Loop Function e di una User Function contemporaneamente). L'Ambiente è definito in ARGoS in alcuni contesti *Arena* ma è sostanzialmente un termine intercambiabile.

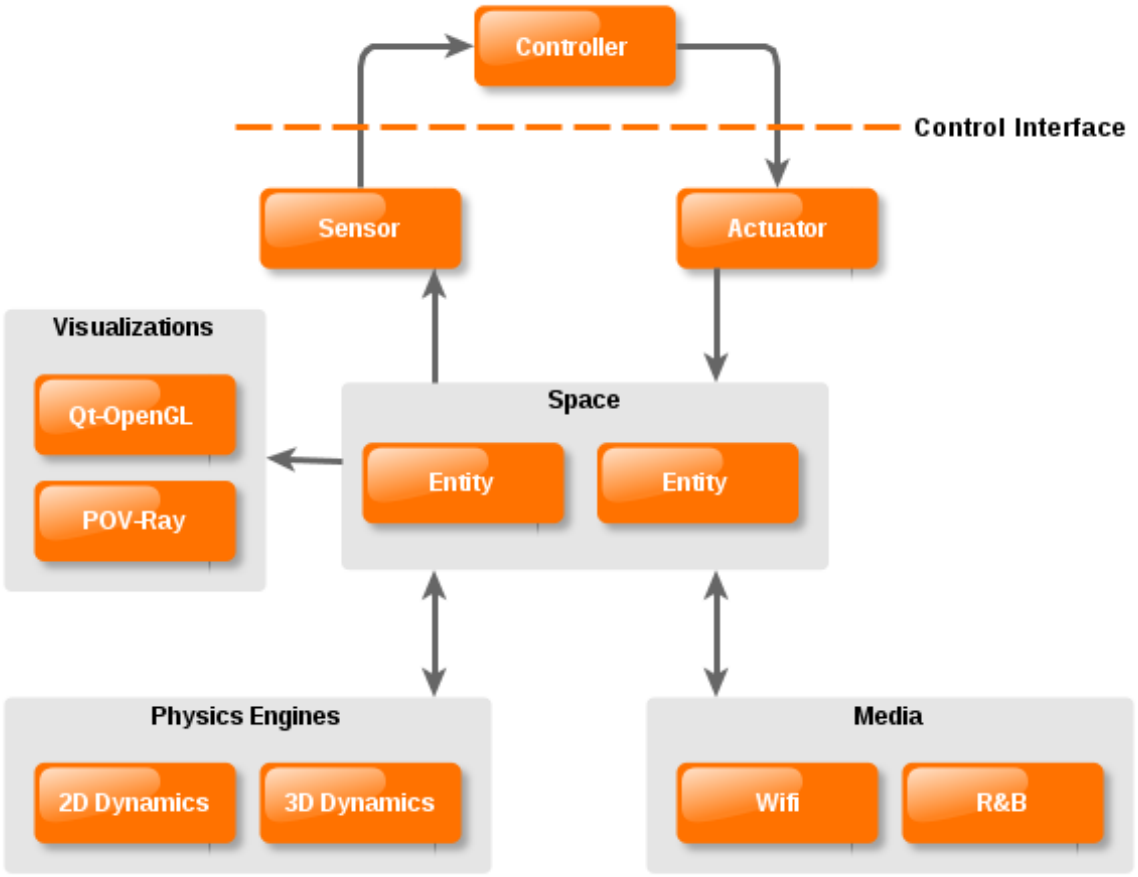

Figura 1: Architettura di ARGoS

3 Modello

In figura 2 si riporta lo schema logico dei componenti implementati per la simulazione. Il *cervello* del drone è suddiviso in due emisferi: un **TaskAllocator** che si occupa di decidere a quale target deve mirare (e quindi dove andare), un **TaskExecutor** che esegue questa volontà decidendo come raggiungere gli obiettivi del drone e come evitare intoppi (per esempio collisioni). Entrambi questi componenti fanno uso di un **Random Number Generator**, del **Positioning Sensor** (che fornisce informazioni circa il proprio orientamento e il proprio posizionamento nell'ambiente) e un **Range And Bearing Sensor**.

Il *Range and Bearing* rappresenta un antenna radio che permette di inviare e ricevere segnali. Questo viene fatto all'interno di ARGoS tramite un buffer di memoria dalla dimensione prefissata (*RAB_size*) all'inizio della simulazione. Il sensore *riceve* questi segnali da una distanza massima prefissata (*RAB_range*), e fornisce al controllore che li processa sia il contenuto del messaggio (*Data*), sia la distanza da cui è stato percepito (*Range*) e l'orientamento locale al sistema di riferimento del drone da cui proviene (*HorizontalBearing e VerticalBearing*). Per non complicare troppo il modello si è deciso di impostare il *RAB_range* ad un valore sufficiente per consentire a tutti i droni di ricevere tutti i segnali da ogni parte dell'Arena. Per come è stato implementato in ARGoS non è possibile aggiornare lo stato dell'attuatore solo quando cambia uno stato locale ma è necessario scrivere un valore ogni volta. Dei dati che devono essere scritti sull'attuatore se ne occupa il Controller, il quale mantiene anche un **log** con alcune informazioni circa lo stato del drone e della simulazione dal punto di vista locale di esso.

Sia il TaskAllocator che il TaskExecutor possiedono uno stato che comanda il loro agire, in particolare in base alle esigenze della simulazione gli stati pozzo (*Idle* e *Arrived*) possono essere utilizzati per terminare la simulazione. Di base essa termina solo quando tutti i droni *ritengono* di aver finito il compito. Questo può avvenire in seguito al superamento del numero massimo di iterazioni (*MAX_ITERATION*), oppure quando il TaskExecutor rileva di essere rimasto *fermo* (o *quasi fermo* oltre una certa soglia di trascurabilità), oppure nel caso in cui si necessita che il TaskAllocator termini dopo aver superato una fase di definizione del target da perseguire. Quest'ultimo scenario (non sfruttato nelle nostre analisi) è utile nel caso in cui si voglia valutare solo l'operato di diverse strategie di allocazione delle attività, (attuate queste si termina la simulazione prematuramente).

Aspetti Teorici Il drone è modellato come un **Agente Isteretico**, in quanto mantiene uno stato interno, percepisce l'ambiente tramite dei sensori e modifica l'ambiente tramite degli attuatori. Inoltre è un **Agente Ibrido** in quanto è la combinazione di due sotto-agenti autonomi tra loro che concorrono a determinare e perseguire gli obiettivi del drone.

Codice Sorgente Il codice di questa implementazione è disponibile nel repository Github **frefolli/argos-experiments** [1]. Per le analisi sui risultati sono stati scritti inoltre degli script Python per elaborare i log prodotti dai controller, diponibili nel repository **frefolli/argos-visualizer** [2] insieme ad un ulteriore simulatore che permette di riprodurre (sebbene in modo più semplice) i log in forma di simulazione grafica. Infine si è fatto uso di una versione modificata di ARGoS per correggere alcuni errori / implementare alcune feature lasciate parzialmente inutilizzabili dallo sviluppatore originale del simulatore. Questa versione modificata è disponibile nel repository **frefolli/argos3** [3].

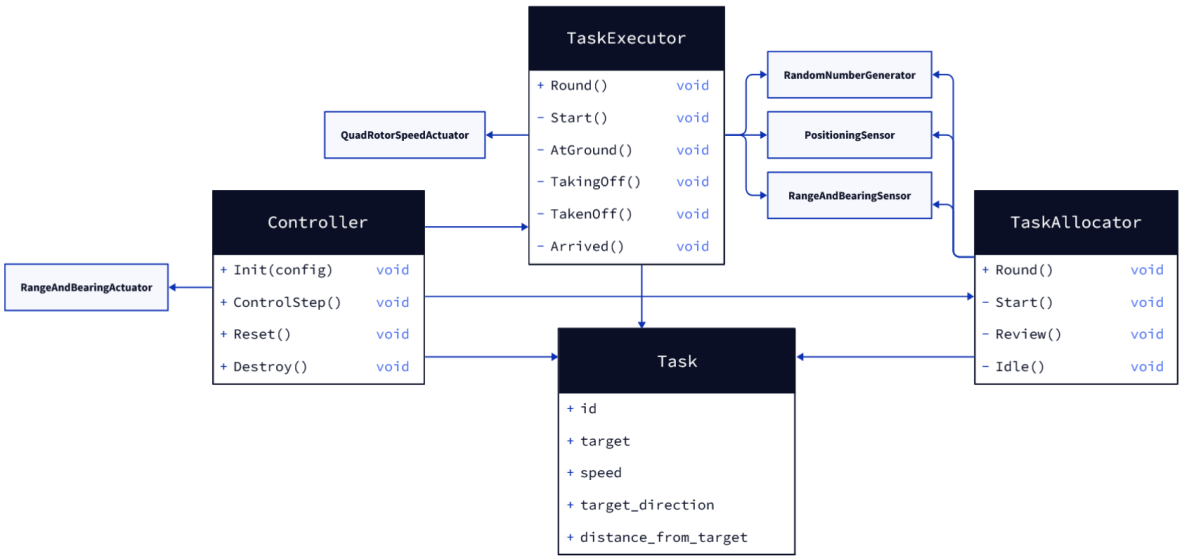

Figura 2: Modello della Simulazione

4 Strategie implementate

4.1 TaskAllocator

Un riassunto dell'implementazione del TaskAllocator è osservabile in figura 3.

4.1.1 Letteratura

La maggior parte dei paper consultati consigliano approcci dove dopo una scelta iniziale (randomica, secondo qualche criterio di vicinanza .. etc) i droni comunicano direttamente o ascoltando le intenzioni dei vicini e prendono delle decisioni su base probabilistica per distribuire uniformemente i droni nei *task* da effettuare. Di conseguenza le nostre implementazioni si sono basate su queste idee generali. Nel nostro caso il task è approcciare uno dei target.

4.1.2 Scelta iniziale del Target

Ogni drone all'inizio della simulazione sceglie un target iniziale, questo può essere fatto scegliendo un target a caso o prendendo il più vicino. Il secondo approccio punta a minimizzare le disanze, ma il secondo (se la distribuzione è uniforme) tende a minimizzare la varianza delle grandezze degli squadroni. Si ricorda che ogni squadrone è definito come i droni che agganciano un target. Dopo la fase iniziale di scelta del target si può decidere di considerare l'allocazione terminata senza ulteriori modifiche o di eseguire qualche iterazione di un algoritmo decentralizzato che tenti equilibrare le composizioni degli squadroni. Questa seconda fase di allocazione verrà chiamata *revisione*.

4.1.3 Fase di revisione della scelta del Target

Durante le iterazioni della fase di revisione è possibile cambiare il proprio target solo se il proprio squadrone ha troppi droni assegnati rispetto a quanti ne dovrebbe avere. Nel caso in cui per un drone sia possibile cambiare il proprio target, abbiamo creato due strategie possibili che può adottare:

1. cambio squadrone scegliendone uno a caso
2. cambio squadrone scegliendo quello più bisognoso di droni.

La rilocazione, possibile solo nel caso in qui in mio squadrone sia in eccesso di droni, può avvenire certamente o avvenire con una certa probabilità. Questa governa sia la probabilità che la distribuzione di droni negli squadroni diventi più equa sia il numero di cambi che vengono effettuati (rendendo più instabile lo stato globale).

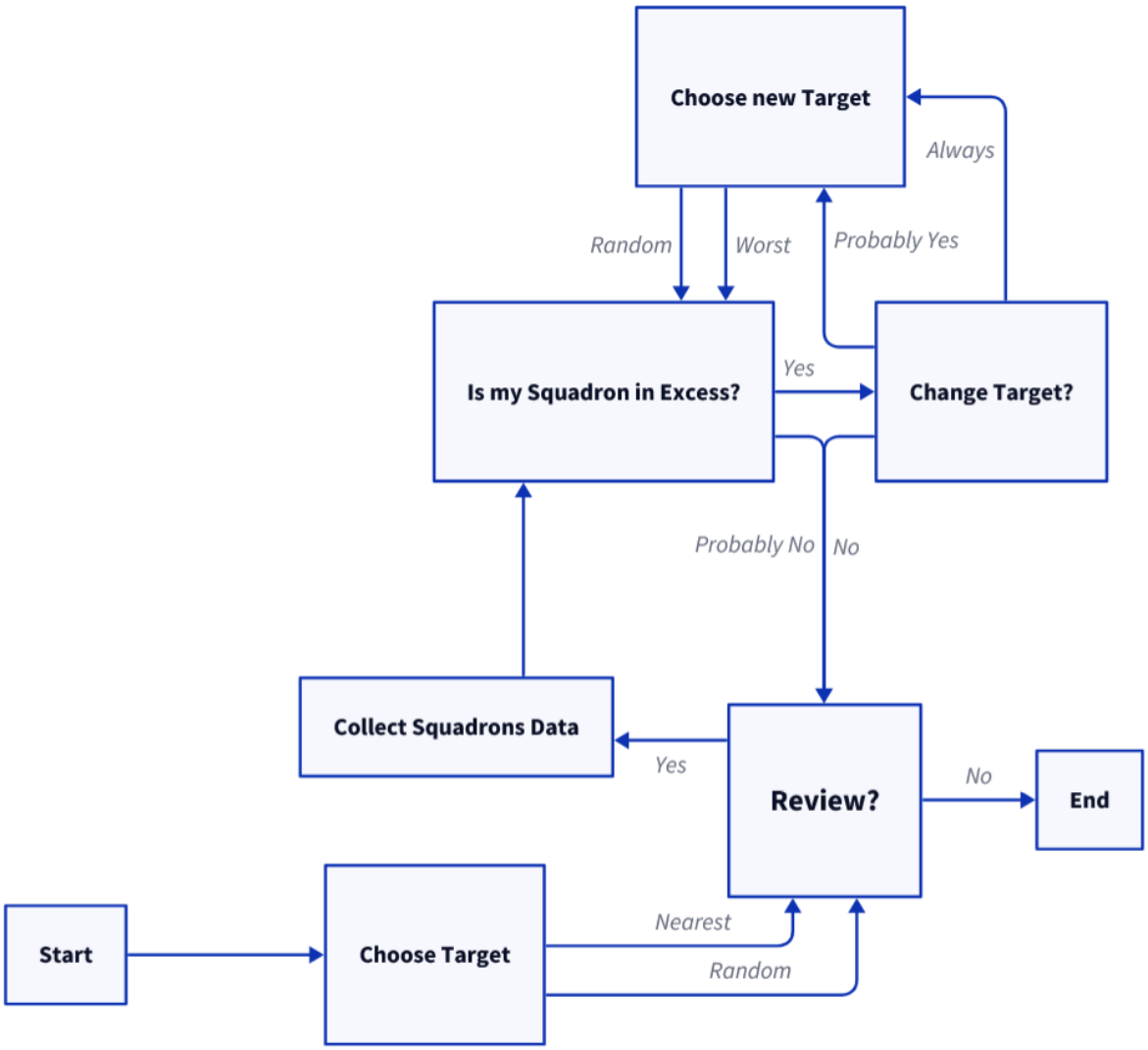

Figura 3: Comportamento del Task Allocator

4.2 TaskExecutor

Un riassunto dell'implementazione del TaskExecutor è osservabile in figura 4.

4.2.1 Letteratura

Molti paper consigliano come algoritmo generale un sistema basato su *forze* dove un drone risente di un'attrazione verso un punto di interesse e di una repulsione nei confronti degli altri droni. Generalmente la magnitudine della forza di repulsione è governata da un *potenziale*, mentre c'è più libertà sul come implementare la forza di attrazione.

4.2.2 Preparazione e Decollo

Dopo una fase iniziale di attesa (si attendono MAX_REVIEWING_SESSIONS=20 tick in attesa che il TaskAllocator prenda le sue decisioni circa il target da perseguire) il drone passa in modalità decollo. La fase di volo può avvenire tramite un iniziale decollo verticale e un successivo volo quasi orizzontale verso il target, oppure tramite un decollo diretto che utilizza il vettore

di movimento diretto verso il target per prendere quota. Il decollo è governato dall'esecutore tramite un audit dei droni nelle vicinanze per dare priorità al decollo dei droni con ID più alto.

4.2.3 Volo

Un drone che ha raggiunto la quota di volo o che decolla direttamente deve eseguire un algoritmo che permette allo stesso di volare in direzione del target ma anche di evitare gli altri droni che ostacolano il percorso. Questo è fatto tramite l'applicazione di forze repulsive tra droni, in particolare tramite i trè potenziali che sono stati implementati: **GP e JP** sono due potenziali ispirati dalla gravità, **LP** è derivato dal potenziale di Lennard Jones.

Detta d la distanza tra due corpi soggetti alla forza repulsiva (i droni), A un moltiplicatore specifico di ogni potenziale utilizzato per ottimizzarne l'intensità e D una distanza media che si vuole mantenere tra due droni, si riportano le formule per ricavare le forze di attrazione:

- $GP(d) = -A_{GP}\frac{|D-d|}{d}$
- $JP(d) = -A_{JP}\frac{D-d}{d^2}$
- $LP(d) = -A_{LP}4(\frac{D}{d}^6 - \frac{D}{d}^{12})$

Si riportano anche i valori dei coefficienti A che abbiamo utilizzato:

- $A_{GP} = 4.0$
- $A_{JP} = 16.0$
- $A_{LP} = 0.2$

4.2.4 Criterio di Arresto

Come detto in precedenza, l'esecutore misura la differenza tra la posizione rilevata correntemente e quella rilevata nel tick precedente, la utilizza per stabilire la velocità con cui si è mosso il drone in precedenza e in base alla magnitudine di essa decide se è fermo o quasi fermo. In particolare noi consideriamo fermo un drone la cui velocità istantanea è inferiore a $10e - 4$.

4.2.5 Rumore Desiderato

Infine abbiamo considerato la possibilità di introdurre del rumore nel vettore della velocità desiderata dall'esecutore per creare dell'instabilità sufficiente a smuovere alcune situazioni di stallo nel volo dei droni che si potrebbero creare (ad esempio quando due droni vogliono andare nella stessa direzione ma nel verso opposto). Per realizzare ciò è fondamentale valutare l'impatto del coefficiente di rumore per evitare che l'instabilità sia tale da non permettere alla simulazione di convergere.

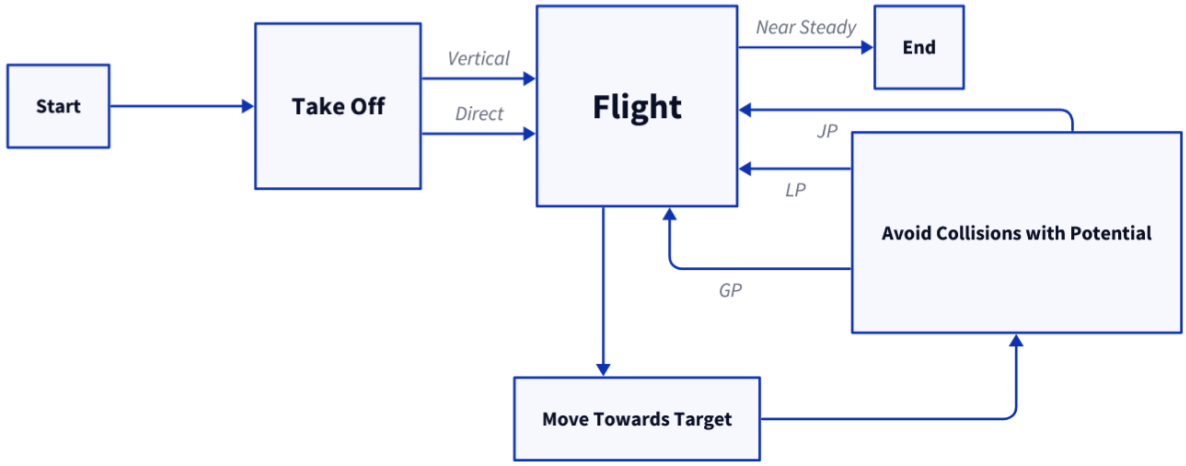

Figura 4: Comportamento del Task Executor

4.3 Esempio Grafico

All'inizio i droni sono tutti a terra e vengono generati 5 target in posizioni randomiche.

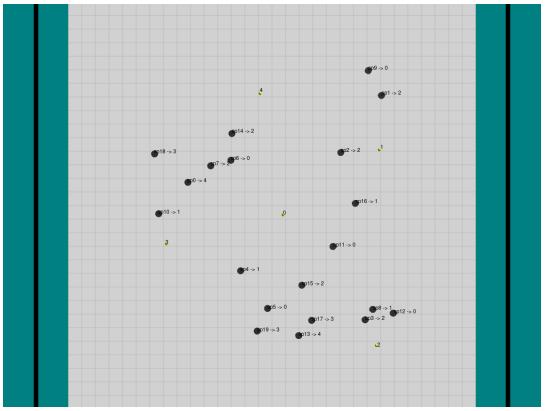

Figura 5: Fase iniziale, vista dall'alto

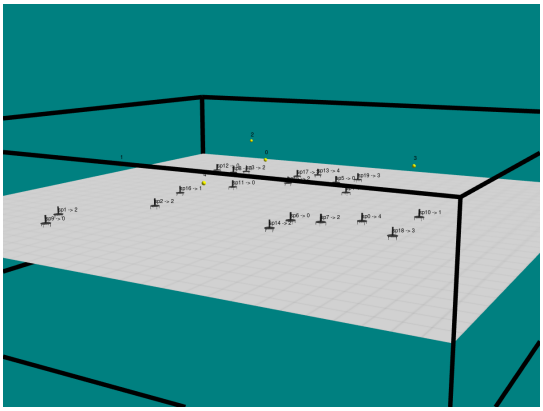

Figura 6: Fase iniziale, vista da sinistra

Dopo le fasi di review, i droni iniziano la fase di ascesa. Alcuni di essi iniziano il decollo, mentre altri rimangono in attesa.

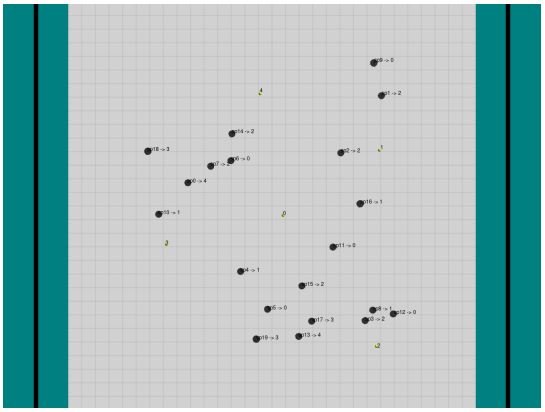

Figura 7: Fase di Ascesa, vista dall'alto

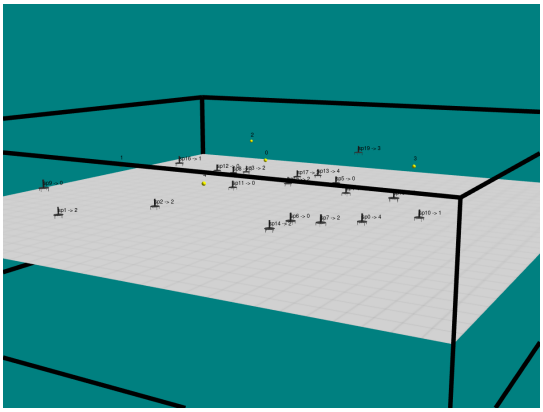

Figura 8: Fase di Ascesa, vista da sinistra

Alla fine della simulazione i droni si sono schierati in formazione attorno ai target. In questo esempio gli squadroni non sono equilibrati. Le strategie implementate in questo progetto devono tendere ad equilibrare la partizione dei droni e ad evitare le collisioni durante il volo.

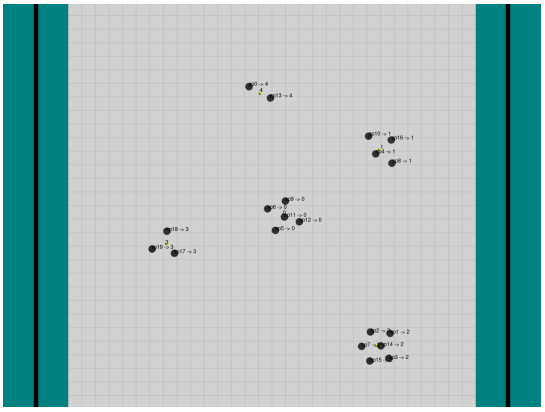

Figura 9: Simulazione terminata, vista dall'alto

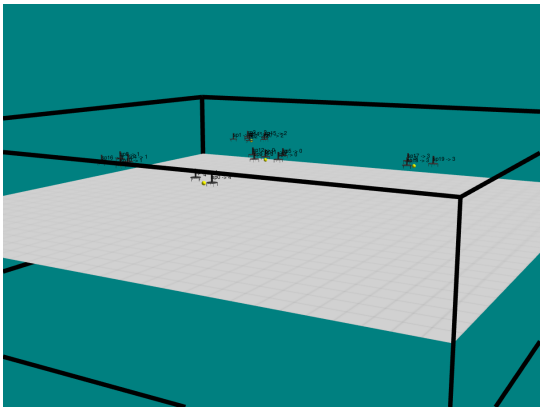

Figura 10: Simulazione terminata, vista da sinistra

5 Esperimenti e Risultati

Sono state condotte delle simulazioni con ARGoS variando in ciascuna la strategia implementata ma mantenendo lo stesso Seed che inizializza il Pseudo Random Number Generator per confrontare direttamente approcci differenti nello stesso scenario. Si è scelto di testare singolarmente il TaskAllocator e il TaskExecutor per ridurre lo spazio delle soluzioni possibili. Ogni esecuzione di simulazione genera dei file di Log che vengono scritti dal Controller di ciascun drone e che ci hanno permesso di produrre alcune metriche utili nell'analisi.

5.1 Prove sul TaskExecutor

In questo primo ciclo di simulazioni abbiamo utilizzato un TaskAllocator *RANDOM__NO_REVIEW*, ovvero che inizializza i target di ogni drone casualmente senza attuare cambi. Questo perchè in questa fase ci interessa osservare come si comportano le strategie di TaskExecutor a parità di seed e di scenario. Quindi si è variato sul potenziale utilizzato nella Collision Avoidance, sul metodo di Decollo e sulla presenza o no del rumore.

5.1.1 Confrontare i Potenziali

Inizialmente ci siamo concentrati sul decollo verticale e in assenza di rumore sul vettore di velocità dell'attuatore variando solo il potenziale utilizzato. In figura 11 è riportata la distanza media dei droni dal proprio target nel tempo.

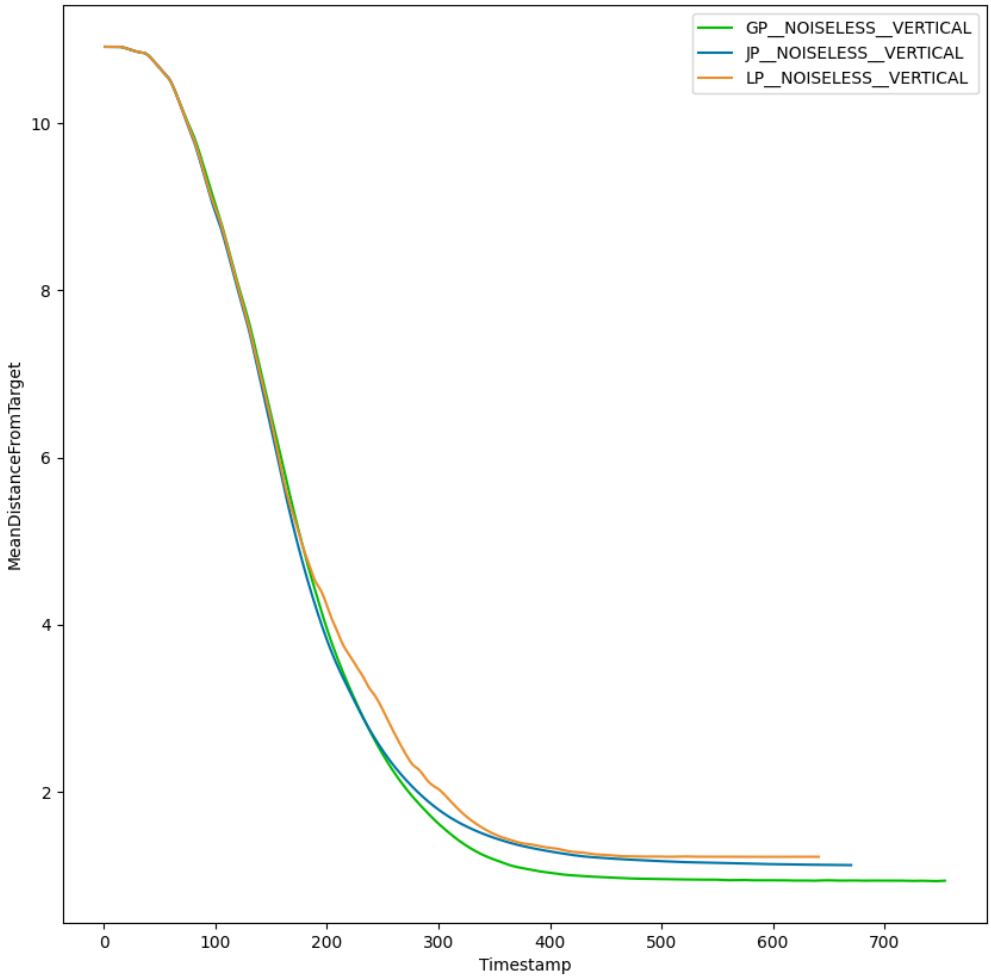

Figura 11: Distanza media dal target nel tempo

Si può notare come i potenziali **GP/JP** hanno curve di discesa più ripide e garantiscono una distanza media inferiore rispetto a **LP**. Da questo si può inferire come con LP la formazione di uno squadrone sia meno compatta (in figura 12 si osserva proprio questo). In generale **LP** dà la possibilità di fissare una distanza minima per i droni e di rispettarla. Lo stesso non vale per gli altri due potenziali. Questa distanza minima tra droni viene mantentuta da LP sia in volo ma anche quando i droni sono assiepati a ridosso del target. Questa può essere vista come un'arma a doppio taglio: certamente le distanze intra-squadrone sono più grandi e quindi il rischio di collisioni diminuisce significativamente, ma i droni quando arriveranno al target saranno e resteranno più separati tra di loro. A seconda del contesto applicativo, se questo aspetto risulta problematico, si può pensare di disattivare/ridurre le forze di repulsione quando siamo vicini al target.

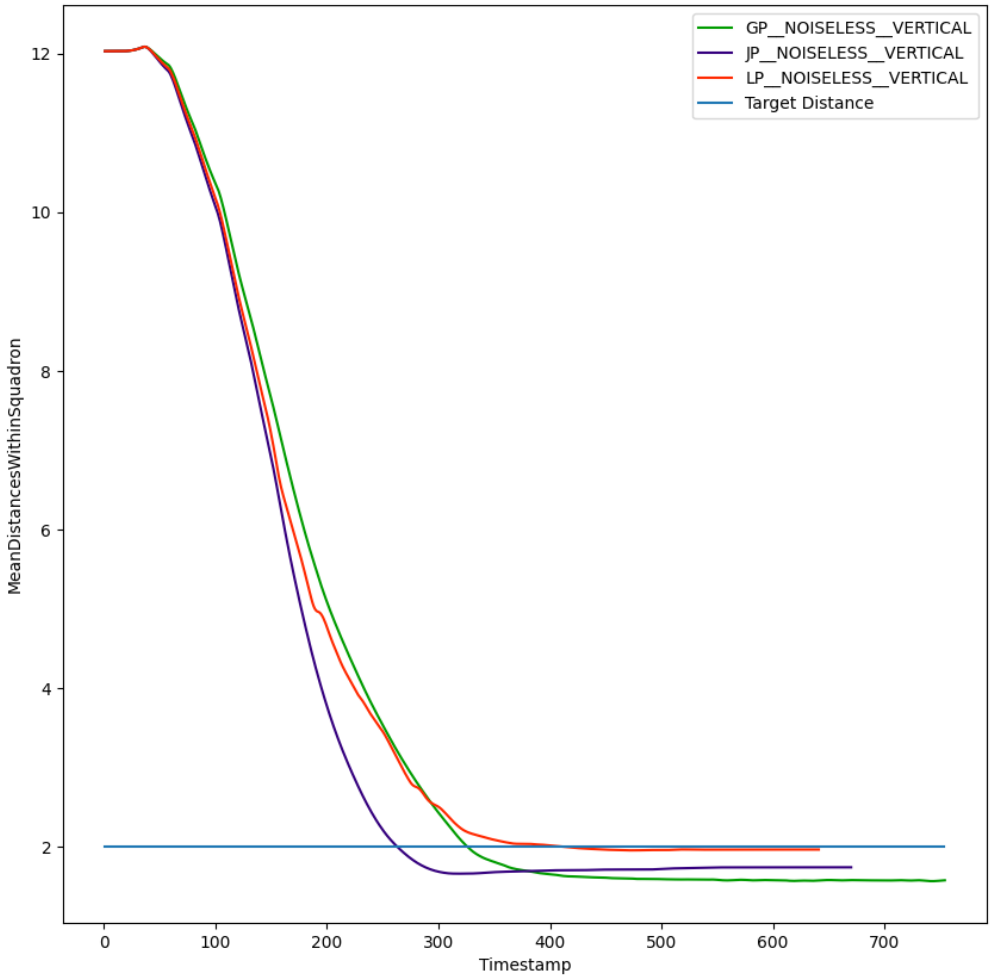

Figura 12: Distanza media dai droni dello stesso squadrone nel tempo

Vediamo come solo **LP** riesca a mantenere la distanza inter-drone sopra una soglia fissata.

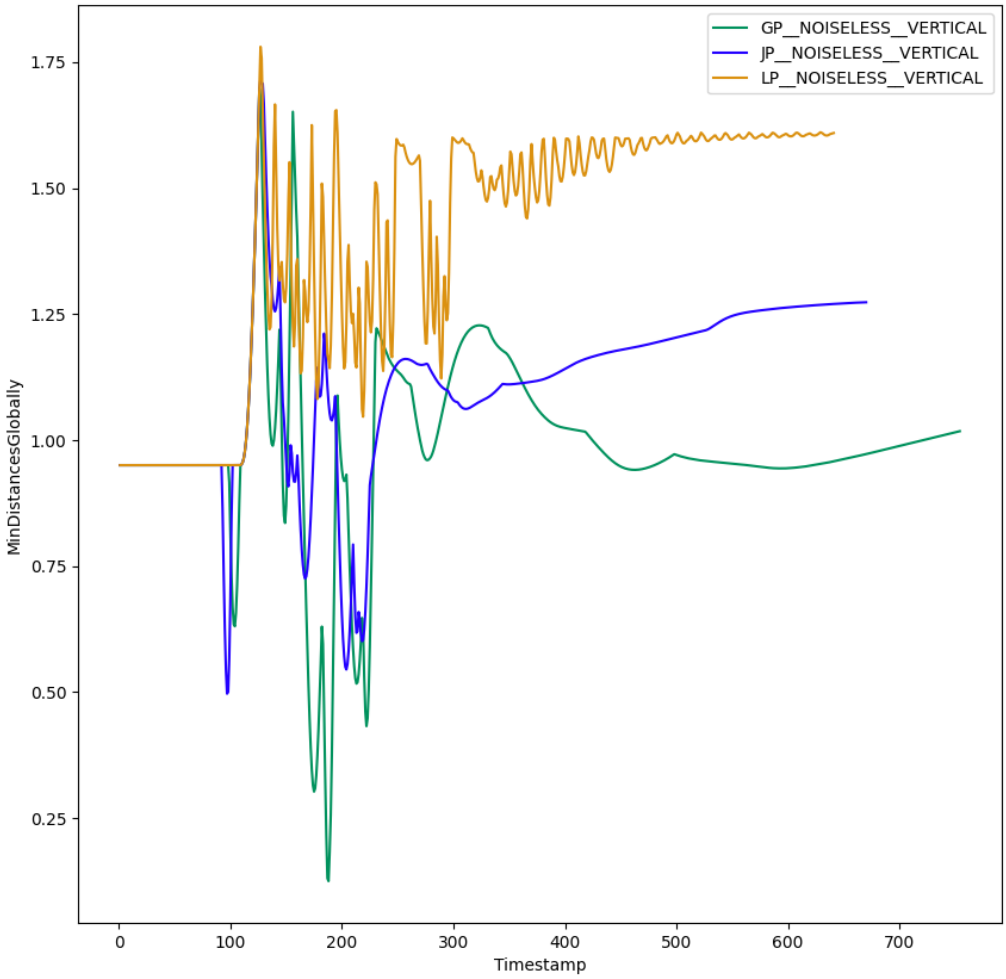

Figura 13: Distanza minima tra droni nel tempo

LP è un potenziale più *reattivo* nella misura in cui a distanze più basse è in grado di applicare una forza di separazione maggiore. Gli altri due sono più rilassati, quindi anche se complessivamente la distanza minima tende in ultima istanza ad essere non di molto inferiore a quella di LP, specie in fase di decollo dei droni possiamo avere diverse occasioni di collisione. Le distanze minime però devono essere interpretate con cautela, perchè se è vero che con GP/JP abbiamo dei picchi negativi in termini di distanza, potrebbe trattarsi di qualche drone già decollato che sta sorvolando un drone in fase di ascesa senza però mai sfiorarlo.

5.1.2 Decollo Diretto

Vediamo ora cosa succede se permettiamo ai droni di saltare la fase di ascesa verticale iniziale e quindi di decollare direttamente nella direzione del target. Nelle figure 14 e 15 sono riportate le medie delle velocità impresse dai droni sugli attuatori nel caso di decollo verticale e diretto. L'effetto principale che possiamo osservare è una diminuzione del tempo totale di formazione dovuto al fatto che il tempo medio di attesa per il decollo dei droni vicini è molto più basso

(dovendo aspettare una generica partenza invece che il raggiungimento di una quota). Inoltre, nonostante si raggiunga una velocità media più alta nel caso del decollo diretto, si noti come i droni rallentino abbastanza presto, e che globalmente ci mettano meno iterazioni a convergere, permettendo quindi ai droni di consumare meno energia rispetto al metodo a decollo verticale.

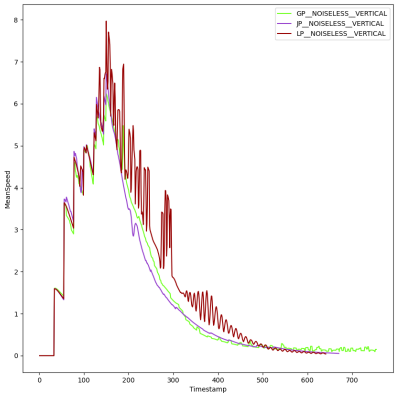

Figura 14: Velocità media voluta dai droni nel tempo con Decollo Verticale

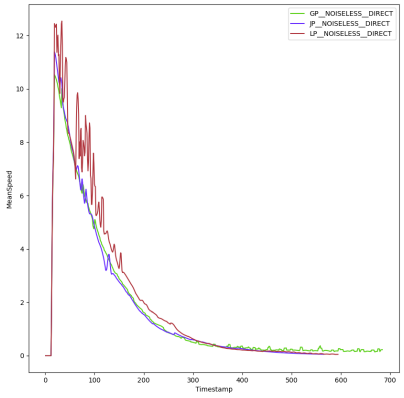

Figura 15: Velocità media voluta dai droni nel tempo con Decollo Diretto

Nelle figure 16 e 17 si riportano affiancate le distanze minime tra droni nei due metodi di decollo. Il decollo verticale è un'arma a doppio taglio perchè le distanze minime tra droni calano sensibilmente aumentando il rischio di collisioni. Nel caso in cui si opti per il decollo diretto risulta particolarmente utile adottare un potenziale reattivo (come LP) che possa separare fortemente i droni in volo.

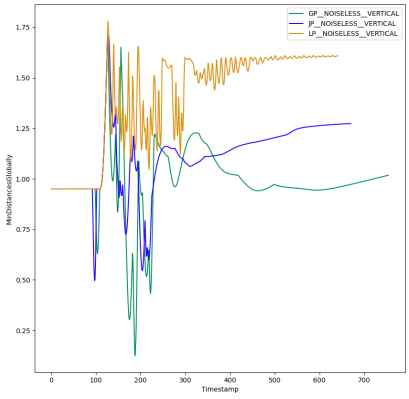

Figura 16: Distanza minima tra droni nel tempo con Decollo Verticale

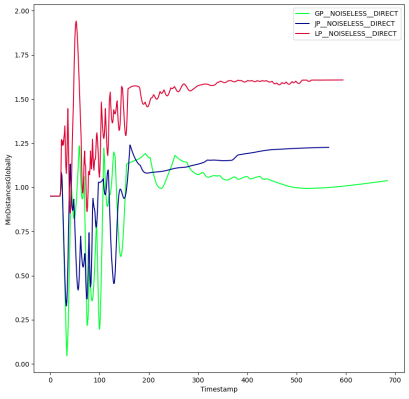

Figura 17: Distanza minima tra droni nel tempo con Decollo Diretto

5.1.3 Aggiunta di Rumore

Come detto precedente il rumore è aggiunto in fase di attuazione del moto principalmente per aggiungere non determinismo in modo da smuovere alcune situazioni di deadlock, ma è anche utile per modellare un sistema a più alto realismo in cui i motori del drone non sono perfetti. L'instabilità creata dal rumore rende le simulazioni più lunghe (ad esempio le figure 14 e 15

versus le figure 18 e 19) perchè è più difficile raggiungere una condizione di equilibrio, però per i parametri del rumore che abbiamo usato, i droni riescono lo stesso a raggiungere i loro target.

Gli andamenti delle metriche mostrate in precedenza si ripresentano invariati (al netto della distorsione data dal rumore). Sebbene si possano evitare deadlock più facilmente (è stato osservato sperimentalmente che in assenza di rumore è possibile che due droni rimangano fermi uno davanti all'altro), il tempo di esecuzione è quasi raddoppiato, determinando quindi un consumo energetico maggiore. In sostanza, nel caso il rumore sia aggiunto volontariamente, la sua magnitudine deve essere commisurata al vantaggio che si ottiene dal non determinismo rispetto allo svantaggio in termini di tempo ed energia. Nel caso in cui il rumore sia significativo risulta più importante scegliere il metodo di decollo che garantisce minor consumo di carburante ma anche più stabilità.

Nelle figure 18 e 19 si riportano le velocità medie dei droni nello scenario con rumore. Si osserva come, a differenza dello scenario senza il rumore, i metodi basati sul decollo verticale garantiscano tempi di convergenza inferiori rispetto al decollo diretto (circa 1000 iterazioni per convergere versus 1400). Guardando nel caso non-rumoroso le figure 14 e 15, si nota che Decollo Verticale senza rumore passa dal convergere in circa 800 iterazioni a circa 1000 se si considera il rumore; mentre Decollo Diretto passa dal convergere in 700 iterazioni (senza rumore) a 1400 se si considera il rumore. Da questo si inferisce come il metodo Decollo Verticale sia meno soggetto al rumore e quindi da preferire nel caso di sistema molto rumoroso. Anche sotto il profilo del consumo energetico Decollo Verticale è meglio di Decollo Diretto, poichè ci mette molto meno tempo a convergere.

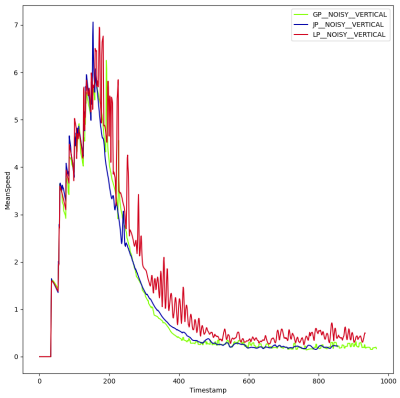

Figura 18: Velocità media voluta dai droni nel tempo con Decollo Verticale e Rumore

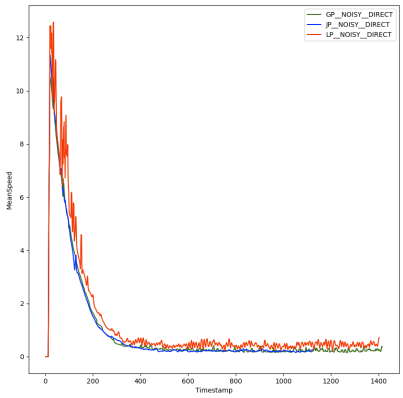

Figura 19: Velocità media voluta dai droni nel tempo con Decollo Diretto e Rumore

5.2 TaskAllocator

Il secondo ciclo di simulazioni riguarda le strategie di Task Allocation, nello specifico in combinazione con un Task Executor con potenziale LP, decollo diretto e senza rumore (si è visto nella sezione precedente come minimizza il tempo di esecuzione mantenendo regolari le distanze tra droni). Nelle figure 20 e 21 si osserva come la strategia di scelta iniziale **Nearest** faccia ottenere distanze da percorrere dai singoli droni più corte rispetto a quanto fa l'inizializzazione Random. Sempre su questo tema, un iniziaizzazione Nearest senza review successive, fa raggiungere il minimo globale per quanto riguarda la distanza media che ogni drone deve percorrere, dato che dopo 200 iterazioni circa i droni sono già arrivati a destinazione.

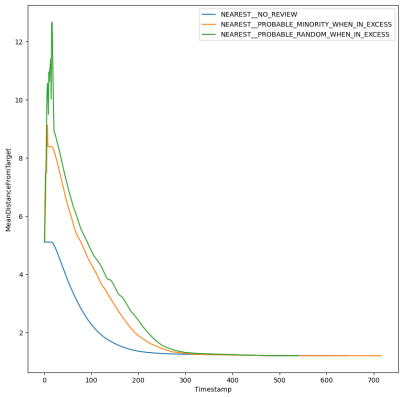

Figura 20: Distanza media dal target nel tempo con scelta iniziale Nearest

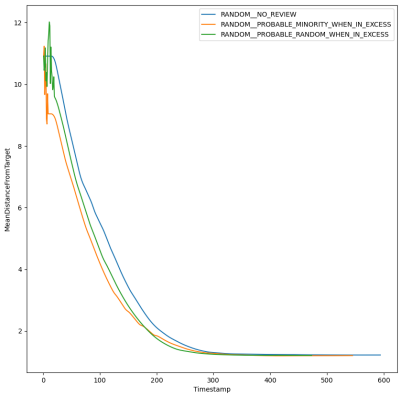

Figura 21: Distanza media dal target nel tempo con scelta iniziale Random

In generale dato che i target vengono assegnati in base alla vicinanza, vale il principio di località, cioè che droni nello stesso squadrone, saranno probabilmente vicini tra di loro, con questo si spiegano i risultati di figura 22.

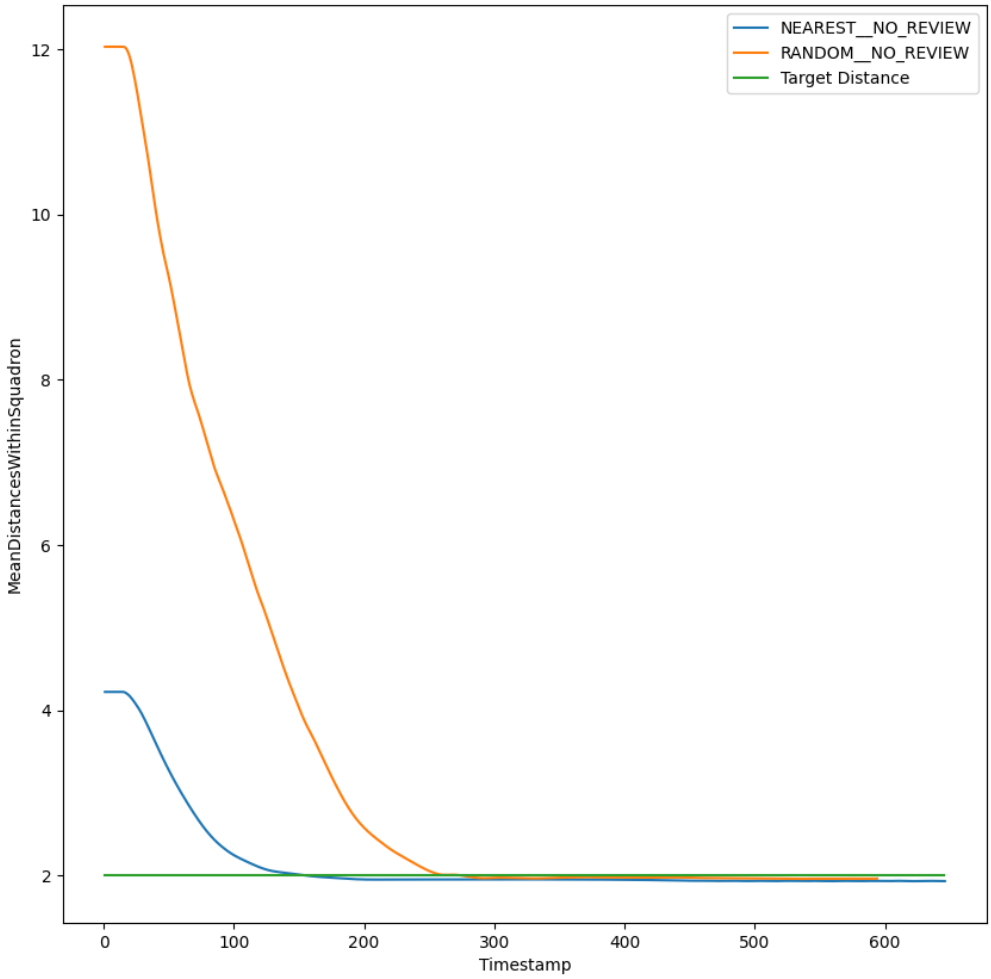

Figura 22: Media delle distanze all'interno dello squadrone nel tempo

Nei metodi dove si fa Review dell'assegnamento ai target (figure 23 e 24), la review comporta per i casi in cui l'assegnamento iniziale era Nearest solo degli svantaggi in termini di distanza media dai bersagli; questo perchè Nearest era ottima come strategia per avere una bassa distanza iniziale di ogni drone dal suo bersaglio. Per i casi in cui l'assegnamento iniziale era Random, si vede come fare Review porti a guadagni in termini di distanza media dai target nel caso in cui si vada ad aggregarsi allo squadrone più bisognoso di droni, mentre se il riassegnamento è casuale, abbiamo risultati più contrastanti (la distanza può anche essere maggiore di quella nel caso noreview).

Si ricorda che il numero di droni che ogni target richiede è uguale per ogni target, quindi interpretiamo negativamente eventuali squilibri di assegnamento tra target.

Parlando ancora della possibilità di cambio di target, si osserva come il criterio che obbliga il drone ad adottare il target meno popolato quando il proprio è in eccesso aiuta a ridurre la varianza di presenza dei droni nei target, quindi lo squilibrio di assegnamento di droni tra target (figura 25), ma solo se la probabilità di cambiare è contenuta, altrimenti si comporta come il criterio di cambio ad uno squadrone casuale (figura 26).

Questo accade perchè tanto più è alta la probabilità di cambio nel metodo Minority, tanti più droni decideranno di cambiare target, e se il target in questione non è casuale ma sempre lo stesso (quello con meno droni rispetto al necessario) si creeranno forti sbilanciamenti. Quindi in queste condizioni una scelta randomica aiuta a distribuire meglio i droni ma è evidente come si possa fare di meglio permettendo a pochi droni (quindi bassa probabilità) di passare allo squadrone più svantaggiato.

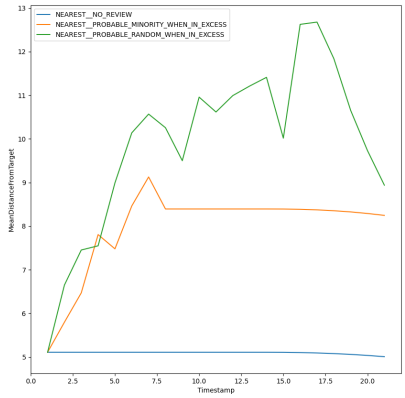

Figura 23: distanza media dei droni dai target con scelta iniziale Nearest

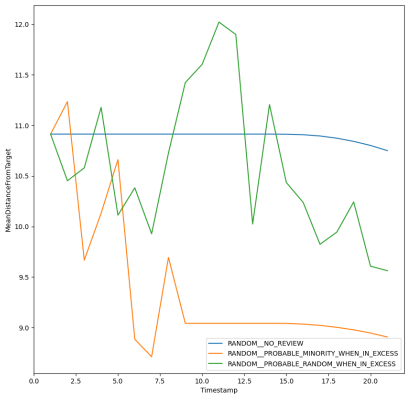

Figura 24: distanza media dei droni dai target con scelta iniziale Random

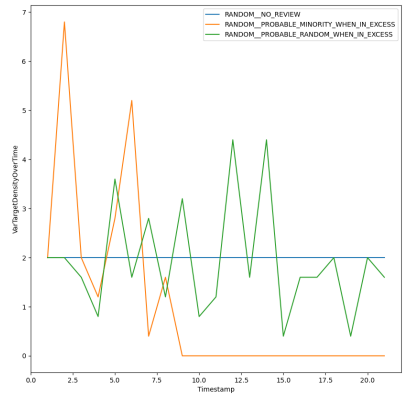

Figura 25: varianza di distribuzione dei droni sui target con probabilità di cambio bassa

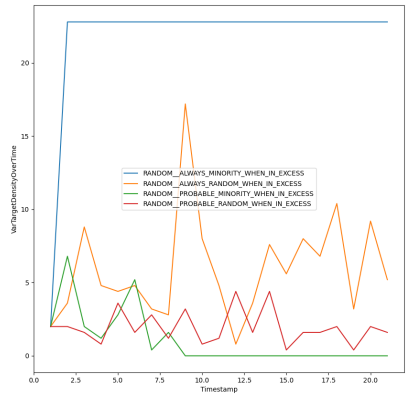

Figura 26: confronto varianza di distribuzione dei droni sui target con probabilità di cambio bassa e alta

6 Limiti del Modello

6.1 ARGoS

ARGoS ha diversi problemi sia ad inserire molte entità nell'ambiente, sia nel simulare in generale un'arena di grandi dimensioni o un numero grande di droni. Per questa ragione i nostri esperimenti si sono concentrati su simulazioni di al massimo 20 droni.

Inoltre, come detto in precedenza, l'implementazione di ARGoS di alcune feature è incompleta, abbozzata o scorretta. Per i droni esso mette a disposizione due attuatori, uno basato sulla posizione (riceve in input una posizione e applica sul drone le forze necessarie a spostarlo nella posizione bersaglio) e uno basato sulla velocità (riceve un vettore di velocità e applica delle forze per raggiungere quella velocità). Il secondo attuatore, quello che abbiamo usato, non solo non era "allacciato" (non veniva mai chiamata la funzione corrispondente nel drone) ma la sua implementazione prevedeva delle spinte assolutamente insufficienti anche solo per spostarlo. Di conseguenza abbiamo dovuto mettere mano al codice per adattare quella parte ad avere almeno la stessa reattività che possiede l'attuatore per posizione (il quale risultava invece funzionante). Il problema di questo aggiustamento risiede principalmente nel fatto che i coefficienti di reattività applicati sono ricavati in modo empirico, quindi se l'attuatore per posizione potesse vantare un certo grado di fedeltà, non si può dire lo stesso di quello basato su velocità.

6.2 Forze di Separazione

Il sistema anti collisione che genera le forze prende in considerazione solo una direzione sul piano orizzontale ma per calcolare l'intensità utilizza una distanza in 3 dimensioni. Questo perchè il modo con cui ARGoS gestisce le rotazioni (specie nei confronti del drone) non è del tutto chiaro e includendo anche la direzione verticale si producevano delle interazioni inconcludenti che sbalzano i droni fuori dalla rotta senza motivo.

6.3 Modellazione delle Collisioni

In generale le collisioni sono modellate come riduzione della distanza tra due droni oltre la soglia minima (diametro di un drone), infatti i droni sono stati pensati come dischi circolari di altezza molto ridotta, ma non abbiamo quantificato questa altezza che potrebbe essere critica per definire le collisioni in aria (per esempio se un drone sorvola un drone in ascesa). Se due droni collidono non è stato definito un comportamento specifico: nella realtà due droni che collidono a bassa velocità ricevono un urto non sufficiente ad abbatterli, ma questo può dipendere anche dalla superficie di impatto (se per esempio l'urto avviene su un rotore) o dalla reattività/fragilità strutturale dei droni stessi.

6.4 Generalità

Eseguire più volte una simulazione con ARGoS con lo stesso *seed* equivale a eseguirlo una volta sola, siccome la sorgente di casualità è interamente generata a partire dal seed.

Avendo fissato il numero di droni, di target e il seed che determina le posizioni di droni e target, i nostri esperimenti possono essere criticati per mancanza di generalità. I risultati tratti da questi sono riferiti a uno scenario con 20 droni, 5 target e 5 droni richiesti da ogni target, e ad una certa disposizione spaziale di target e droni, e per dimostrare che questi sono applicabili a scenari dove variano queste caratteristiche, bisognerebbe fare più esperimenti dove si varia diversi valori di queste variabili.

Ad onor del vero, prima di raggiungere questo setup sperimentale finale abbiamo anche provato con seed diversi ad ogni esperimento (quindi la disposizione di droni e target cambia),

per poi fare la media di 10 simulazioni per ogni strategia analizzata, e i risultati non sono stati poi così diversi.

6.5 Criterio di Arresto

Il criterio di arresto utilizzato dal nostro modello accomoda lo scenario in cui i droni sono arrivati in formazione ma non tiene in considerazione adeguatamente il caso in cui gli stessi finiscano in deadlock. Se due droni sono fermi uno davanti all'altro perchè vorrebbero procedere in direzione l'una opposta all'altra, sarebbero fermi esattamente come se fossero fermi in formazione su un target. Per evitare questa eventualità le simulazioni sono state monitorate graficamente per assicurarsi che fossero valide e in aggiunta si è sperimentata l'aggiunta di rumore al vettore di velocità passato all'attuatore.

Inoltre quando è stato introdotto il decollo diretto, è stato necessario modificare l'algoritmo che verifica il criterio per accomodare questo metodo di decollo siccome per come è fatto il sistema di accelerazione di ARGoS la differenza di posizione per qualche iterazione era tale da far scattare il criterio di arresto.

7 Conclusioni

Sono state eseguite diverse simulazioni che hanno evidenziato aspetti positivi e negativi delle strategie implementate, rispetto anche a diversi scenari (rumore si/rumore no, e un potenziale tra LP, GP e JP).

I droni possono implementare una strategia di selezione del target iniziale tra: una scelta casuale ("strategia Random") o la scelta del target più vicino ("Nearest"). Fatta questa scelta, essa può essere messa in discussione o no, farlo significa che per un certo numero di iterazioni si valuta se il target attuale ha un numero eccessivo di droni assegani a lui o no. Se ha troppi droni assegnati, può, con una certa probabilità, migrare verso uno squadrone casuale o verso quello con il target più bisognoso di droni ("strategia Probable Minority").

Il drone può operare con degli attuatori a cui risce ad imprimere la direzione che vuole o con attuatori soggetti a rumore nel moto che provocano.

I droni sono immersi in un ambiente con delle forze di repulsione tra droni, la cui intesità è modellata da tre potenziali alternativi (LP, JP, GP).

Abbiamo osservato come inizializzare con Nearest i target assegnati ai droni porta a una distanza media da percorrere durante la missione inferiore a quanto farebbe un inizializzazione Random, e a distanze tra droni dello stesso squadrone inferiori al caso Random.

Con assegnamento iniziale Random, vale che per i metodi Probable Minority, tanto più è probabile la rilocazione, tanto più essi generano squadroni squilibrati nel numero di droni che gli appartengono.

Abbiamo visto che in sistemi privi di rumore il Decollo Diretto permette di risparmiare eneriga, metre se è presente rumore allora il Decollo Verticale è molto meglio in questo.

Il potenziale LP, in generale, risulta risultato il più adatto a garantire una distanza minima tra i droni in volo.

Bibliografia

[1] Refolli F. Preziosa A. *ARGoS - Experiments*. https://github.com/frefolli/argos-experiments. 2024.

[2] Refolli F. Preziosa A. *ARGoS - Visualizer*. https://github.com/frefolli/argos-visualizer. 2024.

[3] Refolli F. Preziosa A. *ARGoS 3 - Modified Version*. https://github.com/frefolli/argos3. 2024.

[4] Carlo Pinciroli et al. ≪ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems≫. In: *Swarm Intelligence* 6.4 (2012), pp. 271–295.

[5] James McLurkin e Daniel Yamins. ≪Dynamic Task Assignment in Robot Swarms.≫ In: *Robotics: Science and Systems*. Vol. 8. 2005. Cambridge, USA. 2005.