Università Degli Studi Di Pisa

Tesi Magistrale in

Computer Engineering

Ambiente di simulazione per lo sviluppo di sciami di UAV nella ricerca distribuita di target

Relatori:
Mario G.C.A. Cimino
Gigliola Vaglini

Candidato:
Domenico Minici

19 luglio 2019

Abstract

Questa tesi si concentra sul problema del coordinamento di più UAV per il rilevamento e il tracking di target distribuiti, in diversi contesti tecnologici e ambientali. È stato sviluppato e rilasciato pubblicamente un ambiente di simulazione, basato sulla tecnologia UAV disponibile in commercio e su scenari reali. L'approccio proposto si basa sul concetto di comportamento dello sciame in sistemi multi-agente, cioè un team di UAV autoformato e auto coordinato che si adatta a layout ambientali specifici della missione. La formazione e la coordinazione degli sciami sono ispirati da meccanismi biologici come il floccaggio e la stigmergia. Questi meccanismi, opportunamente combinati, permettono di trovare il giusto equilibrio tra ricerca globale (*exploration*) e ricerca locale (*exploitation*) nell'ambiente. L'adattamento dello sciame può essere basato su algoritmi evolutivi con l'obiettivo di massimizzare il numero di bersagli tracciati durante una missione o di ridurre al minimo il tempo per la scoperta del bersaglio. I risultati sperimentali dimostrano che l'ambiente permette di estendere e superare gli approcci esistenti in letteratura. È stata effettuata un'analisi comparativa, basata su diversi test di simulazione sul banco di prova sviluppato, per identificare i punti chiave da sfruttare e migliorare nei lavori futuri.

Indice

Capitolo 1

Introduzione

Negli ultimi anni, sono state effettuate diverse ricerche orientate allo sviluppo di strumenti e metodi per il monitoraggio e le operazioni all'interno di scenari complessi. Le missioni all'interno di questi contesti sono generalmente classificate come "noiose, sporche e pericolose" (categorizzazione delle "three Ds", dall'inglese dull, dirty and dangerous), in quanto l'accesso a determinate zone risulta limitato, pericoloso o, in alcuni casi, impossibile, sia ad operatori umani che ai mezzi convenzionali.

Piattaforme aeree come gli UAV (Unmanned Aerial Vehicles), spesso chiamati droni, sono oggi la risposta più frequente alle esigenze di tali missioni, grazie ai recenti progressi tecnologici in materia di miniaturizzazione della batteria, di tecnologia di comunicazione, elaborazione e rilevamento [1]. In particolare, la categoria specifica per questa tipologia di operazioni è quella dei mini-UAV, droni di piccole dimensioni equipaggiati con strumenti di self-localization e sensing utilizzati per la raccolta dei dati necessari al completamento dei task prefissati.

L'individuazione e l'identificazione di un obiettivo sono elementi chiave in tutte le operazioni di cui sopra. In alcune di esse, però, anche il tempo di completamento del task ricopre un ruolo fondamentale, basti pensare ad operazioni di ricerca e recupero di particolari target: in questa tipologia di task un'analisi dettagliata di tutto l'ambiente può rappresentare una strategia inefficiente. Un approccio sicuramente più appropriato potrebbe essere quello di effettuare una ricognizione veloce dell'area, con l'unico scopo di identificare i punti chiave da sottomettere, in un secondo momento, ad un'ispezione più accurata. In questo contesto, la qualità degli strumenti di sensing ha un impatto diretto sulle performance dell'intera missione [2], così come

il numero di attori utilizzati: completare tutte le operazioni con un singolo UAV può risultare rischioso (il drone è il nostro single point-of-failure) ed altamente costoso in termini di progettazione, costruzione e manutenzione del velivolo e dell'array di sensori su di esso presente.

L'utilizzo di uno sciame di mini-UAV, ed il conseguente approccio cooperativo che sfrutta gli strumenti di misurazione di tutti i velivoli, può minimizzare l'errore relativo all'identificazione dei target. I maggiori benefici che ne possono derivare sono robustezza, scalabilità e flessibilità e, per tale ragione, è necessario che ogni individuo dello sciame agisca ad un certo livello di autonomia ed esegua operazioni di sensing e comunicazione locali senza un controllo centralizzato o conoscenze globali della missione. L'elemento chiave del funzionamento di uno sciame di droni – in particolare, una delle principali ragioni per cui parliamo di sciame e non di gruppo – è la cooperazione tra gli individui, che permette di completare il task globale attraverso una serie di meccanismi di coordinazione ed esplorazione attuati dai singoli UAV. La principale ispirazione deriva proprio da osservazioni di animali sociali, quali formiche, api, pesci o uccelli, che riescono ad eseguire operazioni complesse di gruppo attraverso poche regole elementari, mostrando una sorta di conoscenza collettiva costituita con delle semplici interazioni locali [3].

Un meccanismo fondamentale di coordinamento dello sciame è la cosiddetta stigmergia [4]. Attraverso questo processo, gli individui lasciano informazioni nell'ambiente in forma di feromoni, sostanze evaporabili diffuse localmente a cui altri individui reagiscono modificando il loro comportamento di conseguenza [5]. Una strategia che permette l'organizzazione dei droni in sciami, invece, è il flocking, un comportamento emergente basato su semplici regole quali allineamento, separazione e coesione [6]. Una forma simulata di feromone, la strategia di flocking ed una serie di meccanismi di evitamento di ostacoli, dunque, possono essere utilizzate per coordinare lo sciame e raggiungere l'obiettivo della missione.

Una volta introdotti gli attori principali del coordinamento di uno sciame di mini-UAV è necessario valutare tali meccanismi attraverso uno strumento di simulazione. Un simulatore di esplorazione dello sciame ha degli obiettivi diversi rispetto ad un simulatore di volo del drone: quest'ultimo si occupa di modellare aspetti di movimento o ambientali; quando trattiamo uno sciame, invece, la modellazione riguarda il sensing e le regole di comportamento che permettono ai singoli individui di collaborare ai fini del completamento di un particolare task. L'obiettivo di questo documento è quello di implementare un simulation testbed, all'interno del quale è possibile integrare diversi algoritmi al fine di valutarne le performance ed attuare un'analisi

comparativa tra gli stessi.

Prima di passare ai dettagli implementativi dell'ambiente di simulazione verrà presentata, nel capitolo successivo, un'analisi sullo stato dell'arte relativo agli algoritmi bioispirati e sulle varie ricerche che hanno portato alle conoscenze odierne. Successivamente, verranno presentate in dettaglio le caratteristiche chiave degli algoritmi selezionati per la comparazione all'interno del testbed. Infine, verrà illustrata l'implementazione del simulatore, insieme all'algoritmo di Differential Evolution utilizzato per l'ottimizzazione parametrica ed ai risultati a contorno della nostra analisi.

Capitolo 2

Stato dell'arte

Molte ricerche sono state sviluppate nel campo della metaeuristica per la ricerca del target utilizzando uno sciame: queste ricerche sono caratterizzate per l'eterogeneità di obiettivi, metodologie e scenari. Alcune indagini recenti (ad esempio, [7]) hanno tentato di creare una tassonomia per discutere le differenze qualitative tra gli approcci proposti. In questa sezione alcuni lavori rilevanti sono brevemente citati e riassunti per il lettore interessato.

Tra gli approcci di ispirazione biologica esistenti in letteratura, stigmergia e flocking sono ampiamente utilizzati per coordinare uno sciame di UAV in compiti di ricerca di target [5]. Gli autori in [4] hanno proposto uno schema di sciame basato sulla stigmergia, in cui feromoni virtuali sono depositati su una mappa feromonica e rilevati dagli agenti. In particolare, le decisioni relative ad azioni e movimenti sono prese dagli agenti camminatori, mentre gli agenti avatar si impegnano a fare stime in assenza di informazioni sul sensore. Lo schema è applicato a molti scenari, compresa l'acquisizione del target. In [8] gli autori progettano una procedura di multi-hop clustering combinata con la stigmergia per fornire una soluzione ottimale ad un insieme di obiettivi, tra cui il rilevamento e il monitoraggio dei target. Inoltre, propongono un modello di feromoni che include feromoni attrattivi e repulsivi, per marcare rispettivamente i target rilevati e le aree visitate. Un'altra applicazione di questo tipo di modello di feromoni è proposta in [9], per missioni simili.

Il flocking è stato proposto per la prima volta da Reynolds [6]. Un esempio è proposto in [10] dove gli autori presentano una strategia di coordinamento decentrato per gli UAV basata sul flocking. Questo meccanismo è utilizzato per mantenere gli

UAV nel campo della comunicazione e per coordinarsi durante i loro task. In [11] viene proposto un gruppo di UAV multipli ad ala fissa che volano ad una distanza relativamente grande l'uno dall'altro. Una strategia di flocking viene applicata anche in [12] dove gli UAV navigano autonomamente in un'area di ricerca.

Diversi lavori di ricerca hanno proposto miglioramenti per tali metaeuristiche biologiche [13]. Queste, a differenza delle euristiche, non sono problem-dependent e possono essere utilizzate come strumenti generali per la risoluzione dei problemi. Ovviamente, una regolazione parametrica è necessaria per adattare una metaeuristica al task in questione ed una taratura di scarsa qualità può portare a risultati imprevedibili. Per esempio, la stigmergia con feromoni persistenti e con un ampio raggio di diffusione può attrarre troppi agenti e portare a ricerche inefficienti. Il flocking con un ampio raggio di visibilità può portare ad una formazione molto rigida. Poiché nei sistemi multi-agente l'interazione degli agenti non è semplicemente deducibile dalle proprietà dei componenti, un'importante tecnica per studiare le proprietà globali e il loro livello di prevedibilità è la simulazione che può essere utilizzata anche quando si propone una variante di una metaeuristica. I lavori che seguono riassumono alcune varianti di flocking e stigmergia utilizzate per i sistemi multi-UAVs.

In questo contesto, in [14] rispetto al modello a feromoni di ispirazione biologica, solitamente attrattivo/ripulsivo, gli autori codificano informazioni specifiche come sapori feromonici. In [15] l'auto-organizzazione di uno sciame di UAV, impiegato in una missione di copertura territoriale, si ottiene consentendo la comunicazione a breve raggio tra UAV attraverso il gossiping. In [16] gli autori propongono una strategia di coordinamento basata sul modello di flocking dei piccioni. Per evitare gli ostacoli, gli individui più alti nella gerarchia del flock sono informati sulla posizione degli ostacoli, fornendo una strategia di pianificazione del percorso basata su un campo di potenziale artificiale. In [17] gli autori propongono diversi approcci basati su tre diversi algoritmi bio-ispirati per effettuare un confronto di metaeuristica applicata ad uno sciame di robot che deve completare una missione con l'obiettivo di minimizzare il consumo energetico.

La particolarità dell'approccio qui proposto è che le metaeuristiche del flocking e della stigmergia generano, insieme a rilevamento ed attuazione, una logica integrata che è parametrizzata nel suo insieme da un'ottimizzazione di Differential Evolution. Un processo di adattamento, infatti, può ricercare la migliore aggregazione delle metaeuristiche secondo la missione specifica e la misura delle prestazioni, che può essere testata nell'ambiente simulato, esplorando soluzioni normalmente impossibili da considerare in una progettazione convenzionale [18], [19].

Ciò è confermato da lavori di ricerca come [20] in cui è necessario un meccanismo di adattamento che regola la coordinazione dello sciame per adattarsi al layout dello scenario attuale. Un altro esempio di coordinamento adattivo è proposto in [21] i cui autori propongono un sistema di controllo per uno sciame di robot coinvolto in un compito di recupero di oggetti. Cambiando la probabilità di passare da una strategia individuale ad un'altra, gli autori mostrano i miglioramenti rilevanti legati all'efficienza dei robot reali e simulati.

Capitolo 3

Analisi

In questo capitolo verranno illustrati gli elementi fondamentali dell'ambiente di simulazione sviluppato. Di seguito è possibile osserverne una rapida descrizione:

- *Ambiente:* area di esplorazione in cui sono posizionati tutti gli elementi;
- *UAV:* entità incaricata di completare il task prefissato dalla missione;
- *Target:* entità che rappresenta l'obiettivo della missione;
- *Ostacolo:* entità che dev'essere rilevata dallo sciame di UAV;
- *Algoritmo bioispirato:* strategia integrata nell'ambiente di simulazione contenente i vari meccanismi di coordinamento dello sciame.

3.1 Ambiente

L'ambiente rappresenta l'area bidimensionale all'interno della quale lo sciame è vincolato a muoversi. L'ambiente risulta discretizzato sia da un punto di vista spaziale che temporale e può essere caratterizzato da obiettivi sia statici che dinamici, in base alla tipologia di missione. La posizione degli obiettivi e degli ostacoli all'interno di quest'area bidimensionale dipende dallo scenario reale utilizzato durante la fase di simulazione.

3.2 UAV: Unmanned Aerial Vehicles

Quando parliamo di UAV - spesso chiamati droni - ci riferiamo alle uniche entità in grado di muoversi in maniera continua all'interno dell'ambiente di esplorazione. Talì entità sono gli attori principali per il completamento della missione: l'organizzazione dipende dall'algoritmo integrato nel simulatore. In generale, il task primario è quello di trovare tutti i target nel minor tempo possibile oppure di trovare il maggior numero possibile di target entro il tempo di autonomia della batteria, evitando sia gli ostacoli che gli altri UAV. Si assume che i droni non conoscano la struttura dell'ambiente e, di conseguenza, non hanno informazioni sulla posizione degli altri UAV impiegati nella missione. Il movimento all'interno dell'area e il coordinamento dello sciame è dettato dall'informazione feromonica rilasciata nell'ambiente e, naturalmente, dalla strategia utilizzata nell'algoritmo bioispirato utilizzato.

3.2.1 Stati di un UAV

Durante lo svolgimento di una missione, ogni UAV può assumere uno dei seguenti stati:

- *Explorer:* un drone che si trova in stato di esplorazione ha lo scopo di saggiare l'area alla ricerca di target;
- *Coordinator:* nel caso in cui la missione lo preveda, il primo UAV ad identificare un targeti diventa coordinatore. Un attore in questo stato ha il compito di richiamare altri droni per intervenire sul target, in accordo con il task previsto dalla missione;
- *Recruited:* nel caso in cui la missione lo preveda, un drone che si trova in stato di esplorazione può essere richiamato da un coordinatore per intervenire sul target, modificando il proprio stato in *reclutato*, in accordo con il task previsto dalla missione;
- *Waiting:* il numero di UAV necessari all'esecuzione di un target dipende dalla missione. Un drone reclutato, che ha raggiunto il coordinatore, modifica il proprio e resta in attesa che tale numero venga raggiunto;
- *Execution:* una volta raggiunto il numero minimo di UAV per intervenire sul target, tutti i droni nell'area del target, coordinatore compreso, modifi-

ca il proprio stato per *eseguire* il target, in accordo con il task previsto dalla missione.

Ogni algoritmo, integrato nel relativo modulo, può prevedere tutti gli stati o un subset di essi. Nelle prossime sezioni saranno definiti, nel dettaglio, i possibili stati che un drone può assumere negli algoritmi presi in esempio.

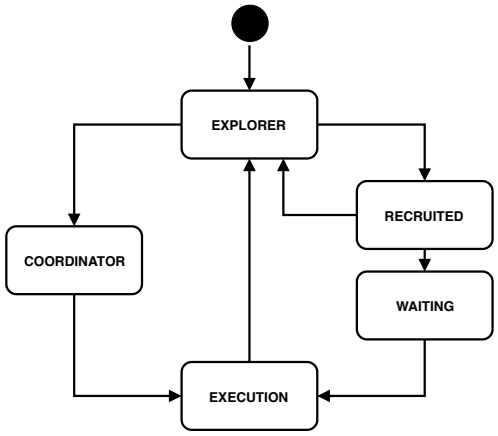

Figura 3.1: Possibili stati di un drone

3.3 Target

I target si trovano in posizioni sconosciute a priori e devono essere scoperti e, se la missione lo prevede, eseguiti dai droni entro il loro tempo di autonomia. Un target può essere, dunque, considerato come *trovato* (found) quando un drone, equipaggiato con una tecnologia di sensing adeguata, si trova in una posizione tale da poterlo rilevare. Un target è da considerare *eseguito* (executed) nel momento in cui un subset dello sciame ha terminato la fase di esecuzione prevista dalla missione.
Alcuni esempi di target potrebbero essere:

- mine antiuomo inesplose;
- discariche abusive;
- principi di incendi;

- ...

3.4 Ostacolo

Un ostacolo è definito come un'entità statica all'interno dell'ambiente di simulazione, che non può essere attraversata dai droni. Costruzioni e alberi rappresentano tipici esempi di ostacoli che possono trovarsi all'interno di un contesto applicativo. Il numero, la forma e la dimensione degli ostacoli dipendono dalla quota di volo di un UAV: tipicamente, maggiore è la quota di volo e minore sarà il numero di ostacoli.

3.5 Algoritmo bioispirato

In questa sezione verranno esaminati alcuni punti chiave fondamentali che caratterizzano tipicamente gli algoritmi bioispirati, oggetto di comparazione all'interno dell'ambiente di simulazione. Per semplificare la comprensione di tali meccanismi, verranno analizzati alcuni algoritmi presenti in letteratura che sono stati integrati all'interno del simulatore per un'analisi comparativa delle performance.

3.5.1 SciaDro

Tale algoritmo è stato inizialmente integrato all'interno del simulatore Sciadro 3.1, pubblicamente rilasciato da Cimino et al. [22]. Nella sezione successiva verranno analizzati i meccanismi fondamentali di questa strategia:

- *Collision avoidance*;
- *Stigmergia*;
- *Flocking*;

3.5.1.1 Modulo di collision avoidance

Tipicamente, come già anticipato, l'ambiente di esplorazione prevede la presenza di ostacoli. Il numero di droni utilizzati per il completamento della missione, inoltre,

potrebbe essere elevato. Di conseguenza, risulta strettamente necessario un meccanismo di collision avoidance al fine di rendere più realistica la simulazione. Il modulo di collision avoidance può essere suddiviso in due sottomoduli:

1. *Modulo di obstacle avoidance:* l'obiettivo principale di questo modulo software è quello di evitare le collisioni tra i droni e gli ostacoli presenti nell'ambiente di esplorazione. In un contesto applicativo reale, questo meccanismo risulta strettamente necessario per ragioni di sicurezza e di costi;

2. *Modulo di overlap avoidance:* l'obiettivo di questo modulo è quello di evitare che si verifichino collisioni (o scontri) tra due o più droni in movimento. Questa condizione deve essere sempre verificata, in particolare quando il numero di droni dello sciame aumenta.

3.5.1.2 Stigmergia

La stigmergia è un meccanismo bio-ispirato usato da formiche, api ed altri insetti sociali, ad esempio durante l'attività di foraggiamento. La stigmergia è un meccanismo emergente, perché entità con limitate capacità computazionali possono organizzarsi in maniera distribuita e robusta.

L'elemento fondamentale della stigmergia è il feromone, ovvero una sostanza chimica in grado di attrarre altre entità nei luoghi in cui viene rilasciata. Il feromone ha una durata limitata nel tempo e dopo un certo periodo scompare a causa dell'evaporazione. Inoltre, un'entità attratta dal feromone è soggetta all'assuefazione olfattiva: dopo un certo tempo l'attrazione esercitata dal feromone perde il suo effetto.

Il meccanismo di stigmergia, così come descritto, è stato progettato, implementato e testato nel simulatore SciaDro. Lo scopo principale è quello di attrarre altre entità su un target di interesse comune, perché, in questo modo, altri target vicini possono essere facilmente scoperti.

Il feromone può essere anche repulsivo: invece di attrarre entità in un'area delimitata, respinge le entità vicine.

3.5.1.3 Flocking

Il flocking è il comportamento esibito quando un gruppo di uccelli, denominato flock, sta facendo attività di foraggiamento oppure è in volo. Il flocking è simile al comportamento di shoaling dei pesci, al comportamento di swarming degli insetti e ai comportamenti di altri animali sociali. La ragione principale per organizzare le entità in flocks sta nel fatto di raggrupparle per rendere più efficiente ed affidabile il processo di scoperta dei target. Infatti, un flock costituito da più droni ha un raggio di sensing più ampio rispetto al singolo drone. Il meccanismo di flocking può essere ulteriormente suddiviso in tre diversi sotto-meccanismi:

1. *Separazione:* il meccanismo di separazione ha l'obiettivo di distanziare i droni, sia per prevenire le collisioni, sia per avere dei flock più ampi così da incrementare la copertura dell'area di sensing;
2. *Allineamento:* il meccanismo di allineamento ha lo scopo di allineare il drone nella direzione media del flock al fine di seguire in modo più accurato gli altri droni della stessa flotta;
3. *Coesione:* questo meccanismo ha lo scopo di recuperare il flock e si attiva quando un'entità è "troppo lontana" dal resto dei compagni. Tale lontananza rappresenta un parametro regolabile all'interno del simulatore.

3.5.1.4 Stati di un drone

Nell'ultima versione dell'algoritmo è stato introdotto un meccanismo che prevede la modifica dello stato del drone, in accordo con determinati eventi che si scatenano durante lo svolgimento della missione. Nel caso specifico, un drone può assumere uno dei seguenti stati, che verranno illustrati in dettaglio nella fase di progettazione:

- *Explorer*;
- *Waiting*;
- *Execution*;

3.5.2 Algortmi PAPER

Gli autori in [17] si sono concentrati sull'applicazione di diverse metaeuristiche di ispirazione biologica per il coordinamento di uno sciame che deve esplorare un'area sconosciuta al fine di trovare e gestire in modo cooperativo alcuni obiettivi distribuiti. Di seguito verranno analizzate prima le metaeuristiche orientate all'esplorazione dell'area e successivamente quelle inerenti il coordinamento dello sciame per la gestione dei target identificati.

In questo caso, un agente può assumere tutti e 5 gli stati descritti nella sezione 3.2.1.

3.5.2.1 Esplorazione

L'obiettivo della strategia di esplorazione è quello di visitare l'area d'interesse della missione in maniera efficiente, minimizzando la possibilità che gli attori visitino più volte una stessa area della regione. A tal fine viene utilizzata una comunicazione basata sull'ambiente (*stigmergia*) ed in particolare viene utilizzato un meccanismo feromonico repulsivo. Durante l'esplorazione, il feromone viene depositato dal drone non appena si raggiunge la nuova cella. In questo modo, l'entità in movimento selezionerà la cella, appartenente al subset di celle immediatamente vicine alla posizione corrente, che presenta la traccia feromonica meno intensa.

L'uso dei feromoni è simile a quello del metodo *Ant Colony Optimization*, ma a differenza delle formiche, in questo caso le celle più interessanti sono quelle senza feromoni o con il più piccolo valore di feromoni. Il feromone depositato si diffonde verso l'esterno, cella per cella, fino ad una certa distanza e la quantità di feromone diminuisce con l'aumentare della distanza dal drone.

3.5.2.2 Strategie di reclutamento

Quando un bersaglio viene identificato, nel caso in cui la missione lo preveda, inizia un processo di reclutamento per poterlo gestire in modo cooperativo. L'attore che ha trovato il target modifica il proprio stato in *coordinator* ed inizia una comunicazione il cui fine è quello di richiamare altri droni all'obiettivo ed eseguire il task previsto dalla missione. Le metaeuristiche di reclutamento sono di tre diversi tipi:

1. *Firefly-based team strategy for robots recruitment (FTS-RR)*: tale metodo è basato sul *Firefly Algorithm* sviluppato da Yang [23], [24]. L'obiettivo di questa strategia è quello di imitare il comportamento lampeggiante delle lucciole. Due importanti fattori sono la variazione dell'intensità luminosa e la formulazione del coefficiente di attrattività. L'intensità della luce diminuisce con il quadrato della distanza, quindi le lucciole avranno visibilità limitata; inoltre l'attrattività è proporzionale alla luminosità quindi per ogni due lucciole lampeggianti la meno brillante si muoverà verso la più luminosa.

2. *Particle swarm optimization for robots recruitment (PSO-RR)*: è una tecnica di ottimizzazione che utilizza una popolazione di agenti multipli [25]. Questa tecnica è stata ispirata dal movimento degli uccelli e dalle loro interazioni con i vicini nello stormo.

3. *Artificial bee colony algorithm for robots recruitment (ABC-RR)*: questo approccio è basato sull'algoritmo ABC sviluppato da Karaboga et al. in [26]. Esso segue il comportamento delle api in cerca di una fonte di nutrimento di qualita. Le api possono essere classificate in tre gruppi: *employed bees*, *onlooker bees* e *scout bees*. Le api *employed* sfruttano le posizioni degli accumuli di cibo, mentre le api *onlooker* sono in attesa di informazioni dalle api *employed* sulla quantità di nettare di tali accumuli. Le api *onlooker* prenderanno la decisione di scegliere una fonte alimentare sulla base di queste informazioni. Una fonte alimentare di qualità superiore avrà una maggiore probabilità di essere selezionata dalle api *onlooker*. Infine, le api *scout* trovano nuove fonti di nutrimento nell'ambiente.

Capitolo 4

Progettazione

Nella fase di progettazione, si sono fatte alcune assunzioni sull'ambiente, sul drone e sulle altre entità coinvolte durante la fase di simulazione. In particolare, tali assunzioni riguardano le caratteristiche principali dell'ambiente e delle entità in esso coinvolte.

4.1 Progettazione dell'ambiente

L'ambiente che viene preso in esame dipende dallo scenario considerato per la simulazione della missione. Esso rappresenta un'area bidimensionale strutturata e limitata, in cui lo spazio ed il tempo risultano entrambi discretizzati.

Lo spazio è simulato attraverso una griglia di NxM celle quadrate, dette *patch*, di lato S. Il tempo è discretizzato in una sequenza di intervalli temporali di durata T detti *tick* definiti come:

$$t_0 + T, t_0 + 2T, \ldots, t_0 + NT$$

L'ambiente simulato contiene i seguenti elementi:

- *Droni*;
- *Target*;

- *Ostacoli*;
- *Feromoni digitali*;

La figura 4.1 mostra una rappresentazione semplificata dell'ambiente di simulazione con gli elementi sopra descritti.

Nell'ambiente, un singolo UAV è rappresentato da un disco con un punta di freccia interna. La punta indica la direzione del drone. Ostacoli e target, solitamente, coprono diverse celle. In figura, ogni cella in cui è presente un ostacolo è nera, mentre ogni cella-target è colorata. Come abbiamo già precedentemente accennato, un target può assumere diversi stati: in questo caso un target *not found* è rappresentato da una cella rossa, un target *found* da una cella verde e con una cella gialla, infine, un target *executed*. L'intensità del colore di bersaglio, quando applicabile (ad esempio, fuoco, gas, ecc.), rappresenta la quantità/presenza del target. Infine, una traccia feromonica è indicata da un gruppo di cellule grigie, dove la scala di grigio varia in base all'intensità dei feromoni.

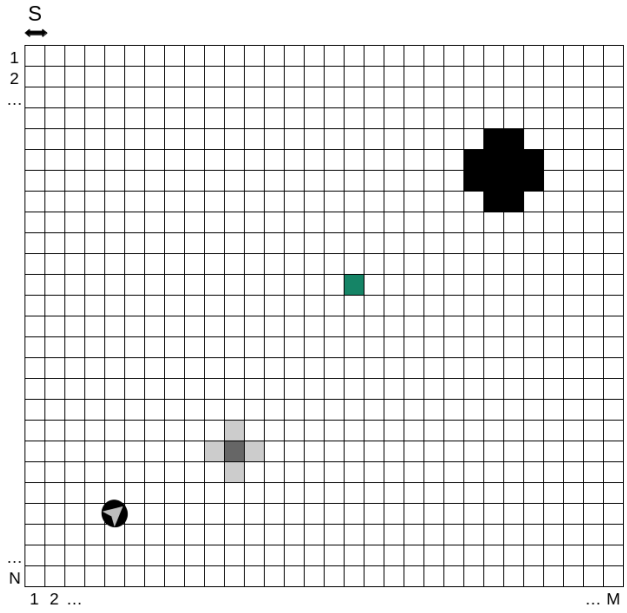

Figura 4.1: Definizione dell'ambiente simulato: (da sinistra a destra) drone, feromone, target ed ostacolo.

Si assumano, per semplicità i seguenti valori:

- S = 1 metro;
- T = 1 secondo;

I valori sopra riportati, naturalmente, possono variare in base alla missione ed allo scenario in cui essa si svolge. In questo caso, l'assunzione è necessaria per una conversione immediata da metri a numero di patch e da secondi a tick.

Supponendo di avere una patch di lato S diverso da un metro, può essere sfruttata la seguente formula di conversione, per determinare il numero di patch a cui corrisponde un metro:

$$1\ m = \frac{1}{S}\ patches \tag{4.1}$$

Allo stesso modo, si può determinare a quanti tick corrisponde un secondo:

$$1\ s = \frac{1}{T}\ ticks \tag{4.2}$$

Dalle relazioni 4.1 e 4.2 è possibile determinare la formula di conversione da *m/s* a *patches/tick*:

$$1\ m/s = (\frac{1}{S})(\frac{1}{T})\ patches/tick$$

Ad ogni tick, il drone effettua rilevamenti di ostacoli, target e feromoni vicini alla sua posizione ed esegue regole comportamentali precise. Lo sciame di droni si muove separatamente, organizzato in diversi flock.

Il blocco fondamentale dello spazio discretizzato simulato è rappresentato dalla cella. Ogni cella (denominata anche patch) ha delle coordinate x-y nello spazio bidimensionale e può contenere un target o un ostacolo. Una caratteristica fondamentale del simulatore è la possibilità di prenotare una patch: tale strategia, infatti, è pensata nel caso in cui è previsto un meccanismo di *collision avoidance*, come nel caso dell'algoritmo SCIADRO.

Per illustrare la struttura di un oggetto *patch*, consideriamo la cella come una classe. Nella figura 4.2 è possibile osservare tale struttura.

Figura 4.2: Struttura della classe Patch.

4.2 Elemento ostacolo

Un ostacolo è un elemento che ostruisce il passaggio di un UAV. Esso ha una posizione statica e occupa una cella all'interno dell'ambiente di esplorazione. L'ostacolo non può essere attraversato dal drone e rappresenta la base per modellare diversi tipi di strutture come, ad esempio, alberi, costruzioni, ecc.

4.3 Elemento target

Un target è un elemento collocato, all'istante t, in una determinata posizione dell'ambiente di simulazione sconosciuta ai droni. La struttura di un target dipende da cosa si vuole modellare, in accordo con il task della missione:

- *Rilevamento di sostanze*, come un gas o un liquido tossico: in questo caso gli elementi sono modellati attraverso una distribuzione gaussiana di target, con media in corrispondenza della sorgente;
- *Identificazione di oggetti*, come ad esempio mine anti-uomo: in questo caso il target è modellato come una singola cella;
- ...

Lo stato di un target è identificato dalla variabile **target.state**, che può assumere i seguenti valori:

- *Found*;
- *Not found*;
- *Executed*.

Perché un target venga lavorato, dev'essere raggiunto da un numero minimo di agenti: questo parametro può essere configurato prima dell'avvio della simulazione e dipende dalla singola missione.

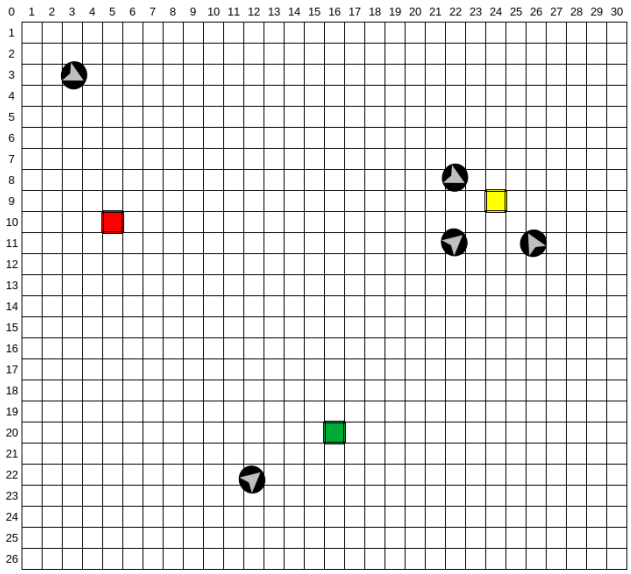

Figura 4.3: Stati di un target: (da sinistra a destra) *Not found, Found, Executed*.

La struttura della classe *Target* è illustrata in figura 4.4.

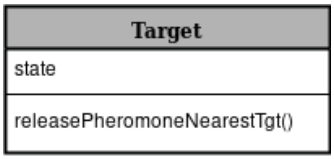

Figura 4.4: Struttura della classe Target.

4.4 Elemento UAV

L'UAV è l'elemento in grado di muoversi in modo continuo all'interno dello spazio bidimensionale di ricerca con l'obiettivo principale di scoprire nuovi target. Inoltre, il drone deve evitare di scontrarsi con gli ostacoli o con altri droni e, insieme a loro, deve coordinarsi e completare il processo di scoperta dei target entro il tempo di autonomia, usando meccanismi di comunicazione diretta o indiretta, in base all'algoritmo integrato nel simulatore.

Gli scontri e le collisioni con altri droni e con gli ostacoli sono, naturalmente, simulati. Per questo motivo, a partire dalla fase di progettazione, questi eventi possono essere denominati "sovrapposizioni tra droni e droni" e "sovrapposizioni tra droni e ostacoli", rispettivamente. Si deve notare che il sensing potrebbe essere influenzato dalla velocità del drone. Esiste una velocità massima per cui la performance della tecnologia di sensing non risulta influenzata dalla velocità del drone: tale velocità è denominata velocità di crociera e risulta minore della velocità massima.

4.5 Algoritmo bioispirato

Alcune caratteristiche degli elementi presenti nel simulatore, come la struttura del drone o del feromone digitale, dipendono dall'algoritmo integrato all'interno dell'ambiente di simulazione. Nella sezione successiva ci soffermeremo sulle metaeuristiche della strategia *Sciadro*. Successivamente, invece, prenderemo in esame gli elementi utilizzati nel gruppo di algoritmi analizzati precedentemente: *FTS*, *PSO* e *ABC*.

4.5.1 SciaDro

In questa sezione entreremo nel dettaglio della progettazione dei vari elementi introdotti nel capitolo 3 (*Analisi*), quali il drone ed i vari meccanismi di flocking, collision avoidance e stigmergia.

4.5.1.1 Progettazione del drone

Il drone può essere modellato come un cerchio in movimento nello spazio bidimensionale. I parametri associati al drone dipendono dalle sue caratteristiche fisiche e architetturali. Lo stato del drone è caratterizzato da:

- Posizione nell'ambiente (**drone.x** e **drone.y**);
- Numero di droni vicini nello sciame (**drone.flockmates**);
- Velocità di crociera (**drone.cruisingSpeed**);
- Velocità corrente (**drone.speed**);
- Direzione corrente (**drone.heading**);
- Velocità angolare corrente (**drone.angularSpeed**);
- Autonomia (**drone.endurance**).

Il drone esegue azioni al fine di:

1. Evitare ostacoli
    - Collision avoidance
2. Coordinarsi con gli altri droni attraverso il meccanismo di flocking
    - Separate
    - Align
    - Cohere
3. Rilasciare il feromone

4. Muoversi nell'ambiente per ricercare i targets

    - Movement
    - Accelerate
    - Decelerate

Al fine di rilevare il target, è stato necessario inserire un codice di sensing. L'area del cono di sensing dipende prevalentemente dal tipo di sensore.

Per le ragioni sopra menzionate, tutti questi parametri devono essere fissati prima di avviare la simulazione.

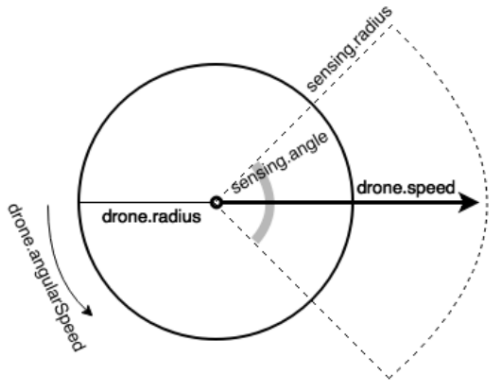

Figura 4.5: Modello parametrico del drone.

La seguente tabella riassume i parametri strutturali che caratterizzano il drone.

Parametro	**Definizione**	**Unità di misura**
drone.radius	Raggio fisico	(m)
sensing.radius	Raggio di sensing	(m)
sensing.angle	Angolo di sensing	(gradi)
drone.cruisingSpeed	Velocità di crociera	(m/s)
drone.speedMax	Velocità massima	(m/s)
drone.speed	Velocità corrente	(m/s)
drone.acceleration	Accelerazione	(m/s^2)
drone.deceleration	Decelerazione	(m/s^2)
drone.endurance	Autonomia	(minuti)
drone.heading	Direzione corrente	(gradi)
drone.accelerationAng	Accelerazione angolare	(rad/s^2)
drone.decelerationAng	Decelerazione angolare	(rad/s^2)
drone.velocityAngularMax	Velocità angolare massima	(rad/s^2)
drone.angularSpeed	Velocità angolare corrente	(rad/s^2)

Tabella 4.1: Parametri strutturali del drone.

Ad ogni ciclo di aggiornamento, il drone assume una posizione specifica nello spazio di ricerca, descritta da una coppia di coordinate. Esso può essere equipaggiato con uno o più sensori e può muoversi sia in modo longitudinale che rotazionale. Il drone, inoltre, è anche in grado di rilevare i target e di rilasciare impronte di feromone per la comunicazione indiretta con gli altri UAV.

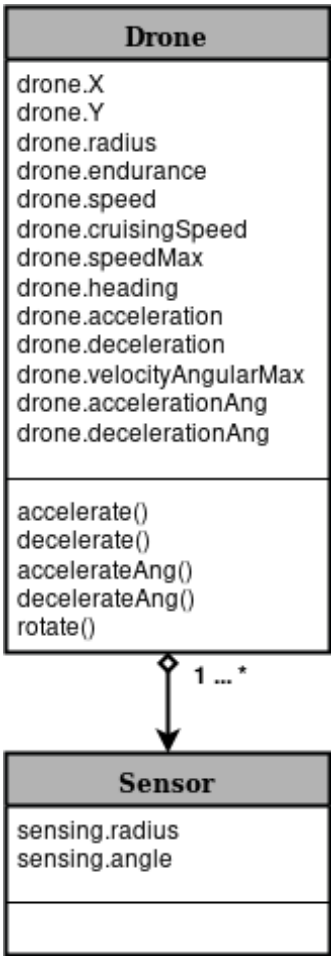

Figura 4.6: Struttura della classe Drone.

4.5.1.2 Progettazione del flocking

Di seguito si illustra la progettazione del meccanismo di flocking descritto nella fase di analisi. Il drone distingue tre diverse aree di prossimità: l'area di separazione (*separate*), l'area di allineamento (*align*) e l'area di coesione (*cohere*). Ciascuna di tali aree ha uno specifico raggio ed è associata ad uno specifico comportamento del drone. Le tre aree di prossimità hanno in comune l'angolo **drone.flocking.angle**.

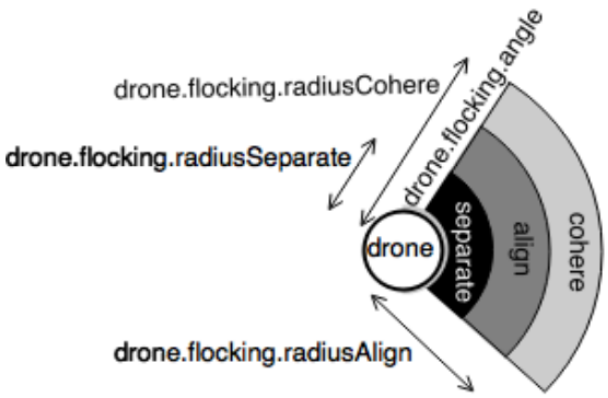

Figura 4.7: Aree del meccanismo di flocking.

I parametri illustrati nella figura 4.7, sono analizzati in dettaglio nella seguente tabella.

Parametro	**Definizione**	**Unità di misura**
drone.flocking.angle	Angolo di visione del flock	(gradi)
drone.flocking.radiusCohere	Raggio di coesione ∈ (drone.flocking.radiusAlign, +∞)	(m)
drone.flocking.maxCohereTurn	Angolo di rotazione massimo nel meccanismo di coesione	(gradi)
drone.flocking.radiusAlign	Raggio di allineamento ∈ (drone.flocking.radiusSeparate, drone.flocking.radiusCohere]	(m)
drone.flocking.maxAlignTurn	Angolo di rotazione massimo nel meccanismo di allineamento	(gradi)
drone.flocking.radiusSeparate	Raggio di separazione ∈ (drone.collisionVision, +∞)	(m)
drone.flocking.maxSeparateTurn	Angolo di rotazione massimo nel meccanismo di separazione	(gradi)
drone.flocking.wiggleVar	Numero casuale ∈ (0,N)	(adimensionale)

Tabella 4.2: Parametri strutturali del flocking.

Comportamento separate: se un drone si trova all'interno dell'area identificata dal raggio **drone.flocking.radiusSeparate**, allora viene attivato il meccanismo di separazione, ovvero una rotazione che porta il drone a separarsi dal resto del flock. Tale rotazione è limitata all'angolo **drone.flocking.maxSeparateTurn**.

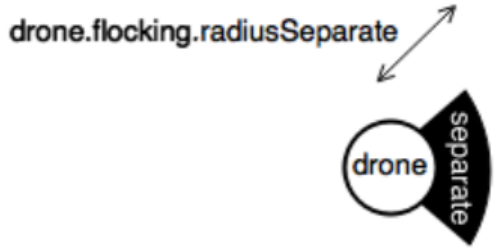

Figura 4.8: Area di riferimento per il comportamento *Separate*.

Comportamento aling: il drone verifica la presenza di altri droni nell'area di align e ruota di un angolo **drone.flocking.alignAngle**, più un angolo casuale **drone.flocking.wiggleVar**, al fine di allinearsi al resto del flock. Tale rotazione è limitata all'angolo **drone.flocking.maxAlignTurn**.

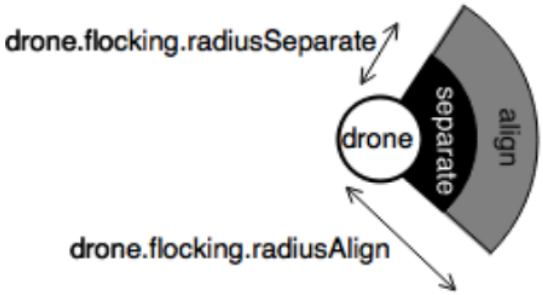

Figura 4.9: Area di riferimento per il comportamento *Align*.

Comportamento cohere: il drone calcola il baricentro dei compagni di flock presenti nell'area di cohere, il cui raggio è **drone.flocking.radiusCohere**, e ruota in quella direzione. La massima rotazione permessa è un angolo pari a **dro-**

ne.flocking.maxCohereTurn, più una quantità **drone.flocking.wiggleVar** casuale.

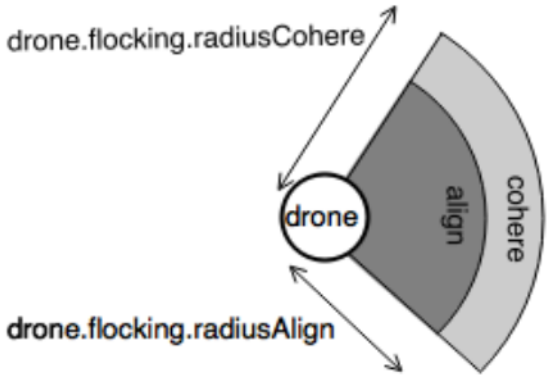

Figura 4.10: Area di riferimento per il comportamento *Cohere*.

Algoritmo di flocking: come già ampiamente illustrato, il meccanismo di flocking è costituito da tre sottomeccanismi a priorità decrescente:

- Separate;
- Align;
- Cohere.

Ad ogni ciclo di aggiornamento (*tick*), può essere eseguito solo uno di questi comportamenti. La figura 4.11 rappresenta il diagramma di attività dell'algoritmo di flocking, in cui viene omessa, per semplicità, la variabile casuale *wiggleVar*.

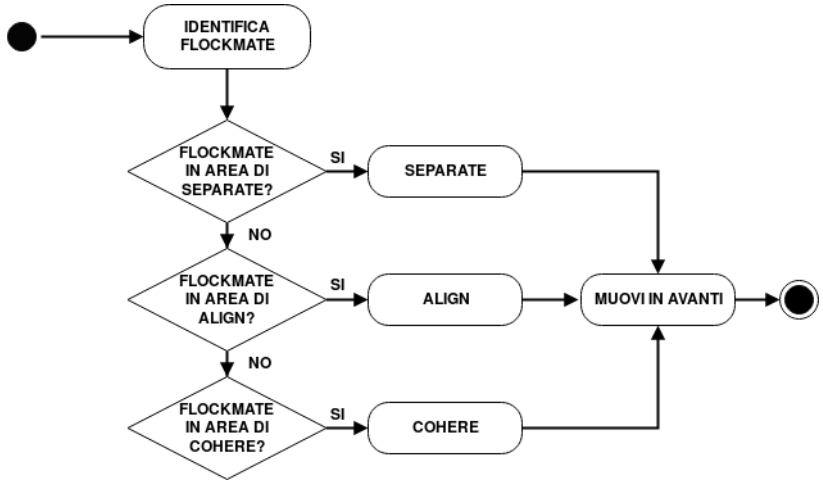

Figura 4.11: Diagramma di attività dell'algoritmo di flocking.

Tutti i parametri di flocking sono fissati prima di avviare la simulazione ed influenzano il movimento del drone e dunque il comportamento del flock.

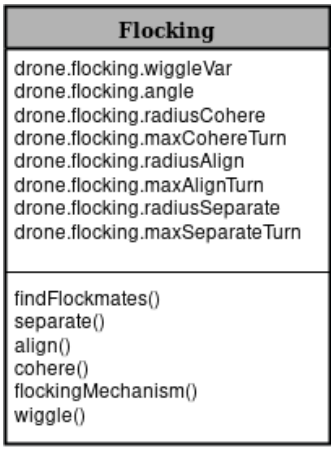

Figura 4.12: Diagramma della classe Flocking.

4.5.1.3 Progettazione del meccanismo di collision avoidance

Il meccanismo di collision avoidance risulta indispensabile, considerato che i droni non devono scontrarsi tra loro o con gli ostacoli presenti all'interno dell'area simulata. Tale meccanismo può essere suddiviso in due sottomeccanismi:

- *Obstacle avoidance*: permette di evitare collisioni tra un drone ed un ostacolo;
- *Overlapping avoidance*: permette di evitare collisioni tra droni.

Modello di obstacle avoidance: il modello segue una semplice regola: trovare il gap tra ostacoli che comporta la minima rotazione del drone verso il gap stesso. La semplicità di questa regola è dovuta al fatto che il drone ha un tempo limitato per riconoscere l'ostacolo, trovare il gap e ruotare verso la sua direzione.

In un contesto reale, il drone può trasportare un array di sensori ad infrarossi in grado di riconoscere le distanze e gli angoli degli ostacoli vicini. L'attività di riconoscimento di un ostacolo è modellata così come fatto per il sensing: un cono con un certo angolo e un certo raggio. Il drone, dopo aver calcolato le distanze e gli angoli degli ostacoli vicini, ricerca il gap che comporta la minore rotazione rispetto alla direzione corrente.

A causa della grande varietà di ostacoli che si possono trovare nei diversi contesti applicativi, sia la dimensione del gap che il raggio e l'angolo di visione degli ostacoli risultano configurabili. Durante la simulazione, questi parametri influenzano il comportamento del drone, perché l'algoritmo di obstacle avoidance dipende dal loro valore.

Parametro	**Definizione**	**Unità di misura**
drone.collision.gapAngle	Minimo angolo che permette al drone di passare attraverso il varco tra due ostacoli $\in$ (0, drone.sight.angleMax)	(gradi)
drone.collision.vision	Raggio di visibilità degli ostacoli	(m)
drone.sight.angleMax	Massimo angolo di visibilità degli ostacoli	(gradi)

Tabella 4.3: Parametri strutturali del meccanismo di obstacle avoidance.

Durante la simulazione, il drone esplora l'ambiente al fine di rilevare i target. Per mantenere la sua operatività ed integrità, l'UAV deve evitare gli ostacoli. Un'idea del funzionamento del meccanismo di obstacle avoidance è mostrato dalla figura 4.13.

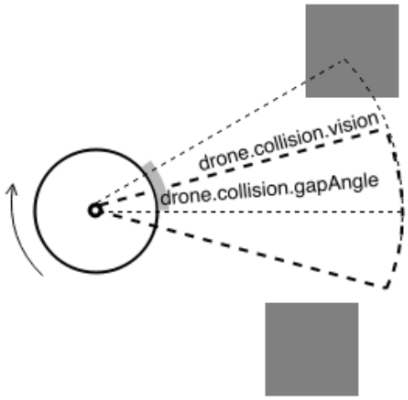

Figura 4.13: Principio di funzionamento del meccanismo di obstacle avoidance.

L'algoritmo opera nel seguente modo:

- Prima di controllare la presenza di ostacoli, il drone deve verificare se è rimasto fermo sulla stessa posizione per un certo numero di ticks consecutivi.
- Se risulta bloccato, significa che l'ostacolo presente di fronte al drone è probabilmente un muro che gli impedisce di avanzare. Al fine di evitare lo stallo, il drone esegue un movimento rotazionale di 90° più una quantità casuale **drone.flocking.wiggleVar**.
- Viceversa, se il drone non è bloccato, allora controlla la presenza di ostacoli.
- Se non ci sono ostacoli, significa che il drone può avanzare senza alcun problema.
- Se sono rilevati degli ostacoli, il drone controlla la presenza del varco tra essi, in riferimento ad un certo angolo di gap.
- Se il varco non viene trovato, allora il drone ruota leggermente in direzione oraria o antioraria (la direzione viene scelta casualmente), al fine di ricercare un altro gap vicino. Questo angolo di rotazione è pari alla metà di **drone.sight.angleMax**. Successivamente, il drone effettua una decelerazione, perché si trova di fronte a ostacoli senza varco.

- Se il varco viene trovato, allora il drone esegue una rotazione verso il gap. Anche in questo caso il drone decelera, perché il varco potrebbe essere stretto e dunque il drone deve attraversarlo con attenzione.

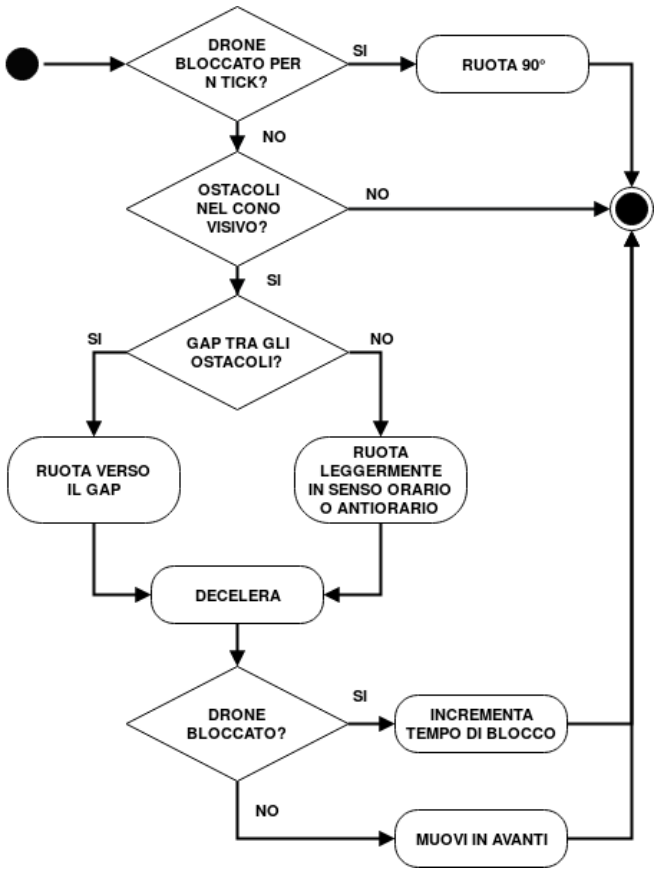

Figura 4.14: Diagramma di attività del meccanismo di obstacle avoidance.

La figura 4.15 illustra il diagramma di classe relativo al meccanismo di obstacle avoidance. Gli attributi **drone.collision.gapAngle**, **drone.collision.vision** e **drone.sight.angleMax** sono configurati prima dell'avvio della simulazione, mentre i restanti due vengono utilizzati per contare il numero di collisioni e l'eventuale tempo di blocco di un drone su una patch.

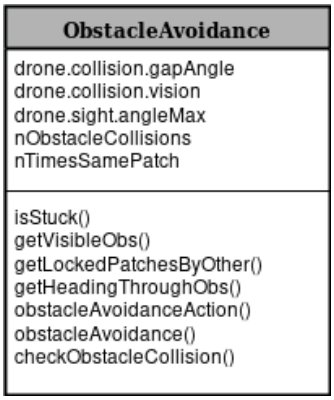

Figura 4.15: Diagramma della classe ObstacleAvoidance.

Modello di overlapping avoidance: il solo meccanismo di obstacle avoidance non è sufficiente per evitare le sovrapposizioni tra i droni, perché un drone non può prevedere i movimenti degli altri droni. Al fine di evitare questo tipo di problema, si è progettato un modello di overlapping avoidance tra droni: conoscendo l'accelerazione, la decelerazione, la velocità massima e la rotazione massima di un drone nell'arco di una finestra temporale, si può calcolare la regione di raggiungibilità (*reachable region*), ovvero la regione che, in termini di patch, il drone potrebbe ricoprire nei prossimi ticks.

In particolare, la strategia per evitare le sovrapposizioni tra i droni prevede, in primo luogo, di schedulare una regione conica di raggiungibilità e, in secondo luogo, di trattare le regioni già schedulate dagli altri droni come ostacoli, applicando il meccanismo di obstacle avoidance. In questo modo, le potenziali regioni in cui un drone può trovarsi nell'immediato futuro non sono accessibili dagli altri droni.

Il meccanismo di overlapping avoidance dei droni risulta altresì configurabile: il numero di droni può variare notevolmente tra i diversi scenari applicativi e le caratteristiche tecniche come velocità massima e massima rotazione risultano molto diverse tra modelli di drone differenti. La regione conica schedulata da un drone per evitare le collisioni è descritta da un certo raggio ed un certo angolo, come riportato nella seguente tabella.

Parametro	Definizione	Unità di misura
drone.reachable.radius	Angolo delle patch schedulate	(gradi)
drone.reachable.angle	Raggio delle patch schedulate	(m)

Tabella 4.4: Parametri strutturali del meccanismo di overlapping avoidance.

L'algoritmo di overlapping avoidance tra UAV, come si evince dalla suddetta descrizione, è suddiviso in due fasi:

1. *Scheduling delle celle*;
2. *Azione di overlapping avoidance.*

Di seguito vengono illustrate tutte le casistiche della fase di prenotazione delle patch:

(a) Nel caso generale, il drone identifica le patch libere sotto la sua posizione ed all'interno del cono di raggiungibilità e le blocca.

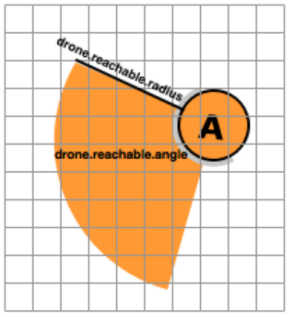

Figura 4.16: Caso generale: prenotazione delle patch nella regione di raggiungibilità.

(b) In altri casi, le regioni di scheduling possono risultare sovrapposte. Nella figura 4.17 l'UAV A blocca le celle nel cono arancione. L'UAV B, di conseguenza, identifica le celle già bloccate da A e prenota solo le patch disponibili in quel preciso istante, ovvero le celle presenti nel cono blu.

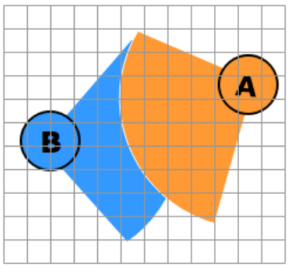

Figura 4.17: Caso di sovrapposizione dello scheduling delle patch.

(c) I casi (a) e (b) potrebbero portare ad un problema di *deadlock*, nell'eventualità in cui un drone A, che si trova esattamente dietro il drone B, prenota le patch davanti a B. In questo caso il drone B non può avanzare, in quanto tutte le celle raggiungibili sono già prenotate.

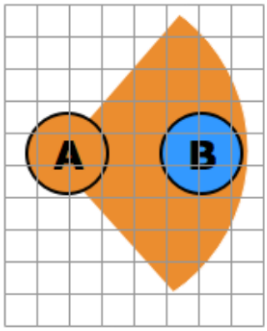

Figura 4.18: Caso di deadlock.

Al fine di evitare tale situazione, risulta necessario introdurre un ulteriore requisito: un drone, prima di eseguire la prenotazione delle celle, deve controllare se all'interno del cono di scheduling è presente un altro UAV. Se il controllo ha esito positivo (è presente un drone nel cono), verrà schedulata solo la cella sottostante il drone che ha effettuato il controllo.

Il diagramma di attività che descrive l'algoritmo di scheduling è mostrato in figura 4.19.

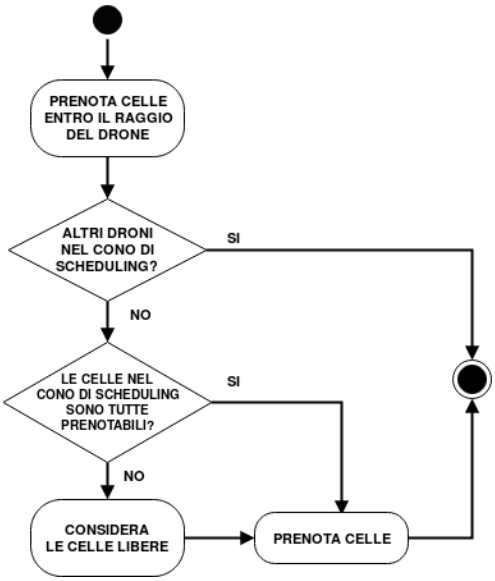

Figura 4.19: Diagramma di attività del meccanismo di obstacle avoidance.

Di seguito, invece, è possibile osservare il diagramma della classe DronesOverlapAvoidance: gli attributi relativi al raggio ed all'angolo del cono di scheduling devono essere configurati prima dell'esecuzione del simulatore e dipendono dalla velocità di crociera assunta dal drone. I rimanenti parametri variano durante la simulazione.

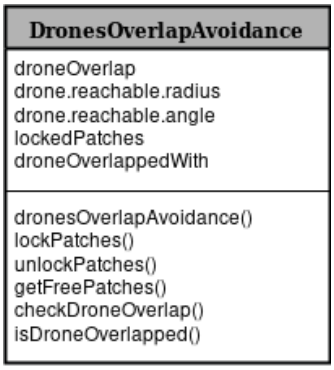

Figura 4.20: Diagramma della classe ObstacleAvoidance.

4.5.1.4 Progettazione dell'impronta di feromone attrattivo

Un drone rilascia un'impronta di feromone attrattivo quando rileva un target, al fine di "richiamare" altri droni attivando il meccanismo di coordinamento basato su stigmergia.

Un'impronta di feromone è modellata come un tronco di cono e tipicamente ricopre più celle dello spazio di ricerca. Più impronte di feromone rilasciate in prossimità spazio-temporale si aggregano a formare una traccia feromonica che evapora nel tempo.

L'impronta è caratterizzata da un raggio centrale (**mark.RadiusTop**), da un raggio periferico (**mark.radiusDown**) e da un tasso di diffusione (**track.diffRate**), mentre la traccia feromonica è caratterizzata da un tasso di evaporazione superficiale (**track.evapRate**).

Le funzioni eseguite sulla traccia feromonica sono:

- *Evaporazione*;
- *Diffusione* (opzionale).

Impronta e traccia feromonica: di seguito viene illustrato il modello dell'impronta e della traccia di feromone attrattivo, insieme ai parametri caratteristici.

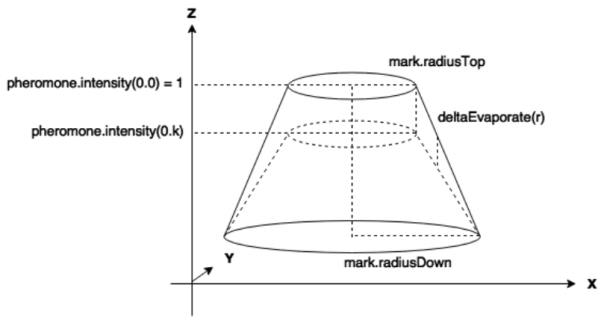

Figura 4.21: Modello tridimensionale dell'impronta di feromone attrattivo.

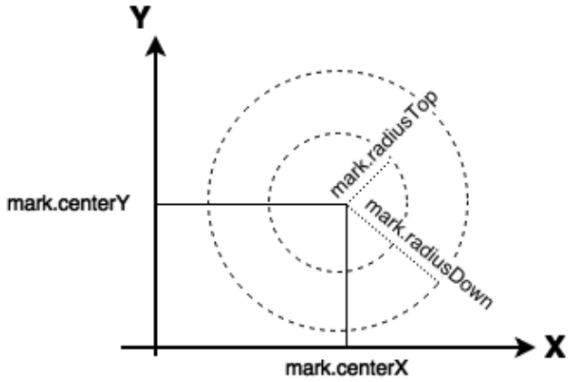

Figura 4.22: Modello bidimensionale dell'impronta di feromone attrattivo.

Parametro	**Definizione**	**Unità di misura**
mark.centerX	Coordinata X del centro di rilascio dell'impronta di feromone	(-)
mark.centerY	Coordinata Y del centro di rilascio dell'impronta di feromone	(-)
mark.radiusTop	Raggio di rilascio iniziale dell'impronta di feromone $\in [0, +\infty)$	(m)
mark.radiusDown	Raggio di rilascio diffuso iniziale dell'impronta di feromone $\in$ [mark.radiusTop, $+\infty$)	(m)
track.evapRate	Tasso di evaporazione della traccia di feromone $\in [0,1]$	(adimensionale)

Tabella 4.5: Parametri strutturali dell'impronta di feromone.

L'intensità di rilascio dell'impronta di feromone attrattivo al tick 0 è pari a:

$$pheromone.intensity(r, 0) = 1\ ,\ r < mark.radiusTop$$

L'intensità decresce a partire da **mark.radiusTop** fino a **mark.radiusDown** in accordo alla seguente espressione:

$$pheromone.intensity(r, k) = pheromone.intensity(0, k) \cdot \frac{r - mark.radiusDown}{mark.radiusTop - mark.radiusDown}$$

dove

$$mark.radiusTop < r < mark.radiusDown,\ k \geq 0$$

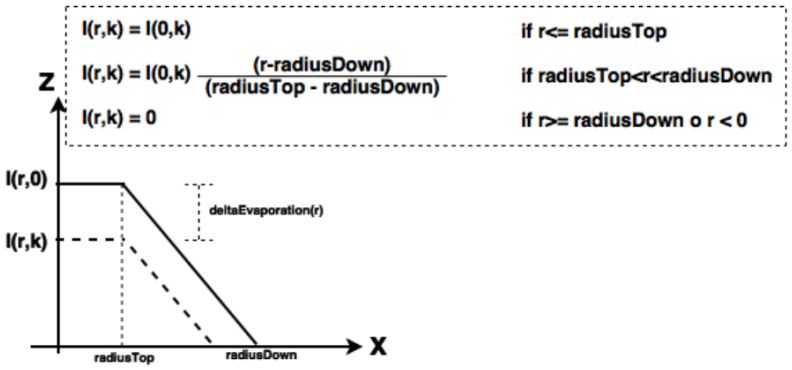

Figura 4.23: Modello dell'intensità di feromone e comportamento di evaporazione.

Comportamento di evaporazione: l'evaporazione rappresenta il decadimento lineare della traccia di feromone nel tempo. Ad ogni ciclo di aggiornamento il feromone evapora di una certa quantità a partire da quando viene rilasciato e dopo un certo tempo, a seconda del valore del tasso di evaporazione, il feromone scompare completamente.

Dal momento che il tempo simulato è discretizzato e l'evaporazione è lineare, ad ogni ciclo di aggiornamento viene sottratta da ogni patch sempre la stessa quantità di feromone. Ad esempio, se il tasso di evaporazione è pari a 0.1, dopo 10 ticks, il feromone scompare completamente indipendentemente dalla sua intensità iniziale.

L'espressione matematica che descrive l'evaporazione feromonica è la seguente:

$$pheromone.intensity(r, k) = pheromone.intensity(r, k-1) \; - \; deltaEvaporation(r)$$

dove

- $deltaEvaporation(r) = evapRate \cdot pheromone.intensity(r, 0)$, ovvero la quantità di feromone che evapora ad ogni tick;
- $pheromone.intensity(r, k)$ rappresenta l'intensità dell'impronta feromonica in funzione della distanza r dal centro di rilascio e del numero di tick k;
- $pheromone.intensity(r, 0)$ rappresenta l'intensità dell'impronta feromonica al momento del rilascio in funzione della distanza r dal centro di rilascio.

Meccanismo di coordinamento basato su stigmergia: come già evidenziato in precedenza, la stigmergia è un meccanismo emergente. Il meccanismo di coordinamento basato su stigmergia segue due semplici regole:

- se un drone rileva un target, vi rilascia sopra un'impronta feromonica;
- se un drone rileva un'impronta feromonica, segue la direzione corrispondente al picco di intensità.

Tale meccanismo ha la potenzialità di attrarre gruppi di droni che sono posizionati in prossimità spazio-temporale rispetto alla traccia feromonica.

Assuefazione olfattiva: un altro importante meccanismo da evidenziare è rappresentato dall'assuefazione olfattiva (*olfactory habituation*). Un'agente, dopo aver rilevato del feromone per un certo numero di ticks, ne diventa assuefatto, ovvero non è più in grado di percepirne la presenza e, di conseguenza, non ne viene influenzato.

L'assuefazione olfattiva risulta importante per due ragioni:

- evita di attrarre un numero eccessivo di agenti sullo stesso target;
- evita di rimanere nei pressi dello stesso target per troppo tempo.

Questi due aspetti permettono di ridurre la probabilità di collisioni ed il tempo impiegato nella ricerca di target in prossimità spaziale.

L'assuefazione olfattiva è caratterizzata da un parametro che esprime la quantità di tempo entro la quale un drone deve annusare il feromone prima di ignorarlo. Questo è un parametro configurabile in modo da poter essere modificato in funzione dello specifico contesto applicativo.

Il seguente diagramma di attività illustra il meccanismo di funzionamento dell'assuefazione olfattiva.

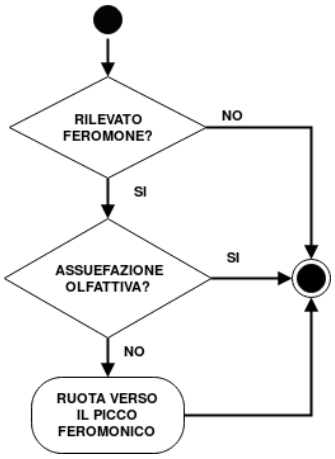

Figura 4.24: Diagramma di attività relativo al funzionamento dell'assuefazione olfattiva.

Di seguito, invece, sono evidenziati tutti gli attributi ed i metodi relativi al feromone attrattivo. Tali attributi, eccetto **mark.deltaEvaporate**, devono essere configurati prima dell'inizio della simulazione.

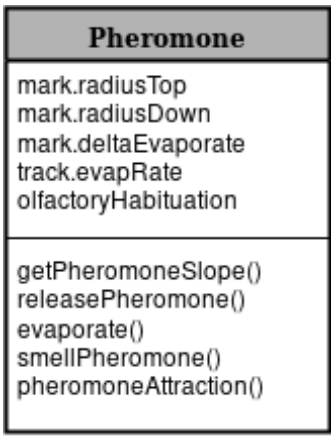

Figura 4.25: Diagramma della classe Pheromone.

4.5.1.5 Progettazione del meccanismo di rilevamento ed esecuzione dei target

Come già evidenziato nella fase di analisi, il drone è equipaggiato con uno o più sensori. Si suppone che un sensore sia ideale e la sua copertura sia modellata attraverso un cono di sensing, con un angolo e un raggio definiti a priori.

Durante l'esplorazione, possiamo avere tre diversi casi, in relazione al rilevamento dei target:

- Nessun target rilevato: il drone prosegue normalmente la sua esplorazione;
- Un solo target rilevato: il drone rilascia un'impronta feromonica centrata sulle coordinate dell'obiettivo;
- Più target rilevati: invece di rilasciare più impronte di feromone, il drone ne rilascia un sola in corrispondenza del target più vicino, come evidenziato nella figura 4.26. In questo modo, il drone rilascerà le impronte feromoniche sugli altri target nei tick immediatamente successivi.

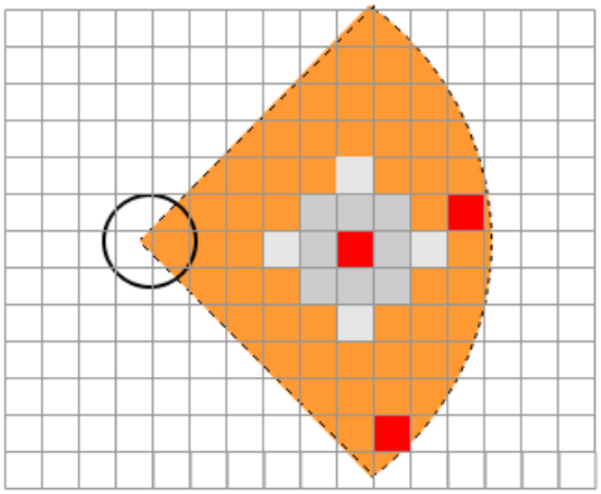

Figura 4.26: Meccanismo di rilascio del feromone in caso di rilevamento multiplo di target.

4.5.2 Algortmi PAPER

Nella fase di analisi abbiamo introdotto la strategia adottata dagli autori di [17], i quali hanno applicato tre diversi algoritmi bio-ispirati per investigare i comportamenti emergenti di uno sciame di robot. La missione di questi agenti consiste nel rilevare dei target distribuiti in maniera uniforme in un'area limitata, attraverso delle metaeuristiche di esplorazione e reclutamento. All'interno del modulo responsabile di ospitare gli algoritmi integrati per un'analisi comparativa, in ogni caso, gli attori non sono dei robot ma nuovamente dei droni, che mantengono molte delle caratteristiche analizzate nelle sezioni precedenti.

Per una migliore comprensione dell'argomento passeremo prima in rassegna i punti chiave dell'algoritmo di esplorazione e successivamente ci concentreremo sugli algoritmi di reclutamento precedentemente introdotti.

4.5.2.1 Strategia di esplorazione

L'obiettivo dei droni, in questa prima fase, è quello di rilevare i target distribuiti nell'ambiente. Per ottimizzare la ricerca di questi obiettivi, gli autori hanno introdotto un meccanismo stigmergico legato ad un feromone repulsivo. In questo modo, un drone che deve selezionare la cella successiva analizza prima il subset di patch ad esso vicine (o *neighborhood*) e, successivamente, seleziona la cella che presenta la traccia feromonica di minore intensità.

Diffusione del feromone: il feromone viene depositato dal drone non appena si muove in una nuova posizione. La struttura del feromone prevede la diffusione dello stesso entro un raggio R_S. Ogni cella ha un livello feromonico iniziale uguale a 0, a voler rappresentare che la patch non è stata mai visitata.

Ipotizziamo che un drone k, all'istante t, si trova nella cella di coordinate (x_k,y_k). La quantità di feromone depositata sulle celle di coordinata (x,y), all'interno dell'area descritta dal raggio R_S, è data da:

$$\Delta\tau_c^{k,t} = \begin{cases} \Delta\tau_0 e^{\frac{-r_{kc}}{a_1}} - \frac{\epsilon}{a_2} & \text{per } r_{kc} \leq R_S \\ 0 & \text{altrimenti} \end{cases} \tag{4.3}$$

Dove:

- r_{kc} è la distanza tra il drone k e la cella c ed è definita come:

$$r_{kc} = \sqrt{(x_k^t - x)^2 + (y_k^t - y)^2} \tag{4.4}$$

- $\Delta\tau_0$ rappresenta la quantità iniziale di feromone che un agente rilascia sulla cella su cui è posizionato;
- $\epsilon \in (0, 1)$ è una variabile euristica atta a modellare il rumore;
- a_1 e a_2 sono costanti utilizzate per ridurre o aumentare gli effetti di feromone e rumore, rispettivamente.

Più droni possono depositare feromone sulla stessa patch. La quantità totale di feromone all'interno di una cella sarà data dalla somma algebrica delle quantità depositate dai vari agenti.

É previsto un meccanismo di evaporazione della traccia feromonica che dipende da un parametro configurabile, detto *tasso di evaporazione* ρ. La quantità di feromone evaporata in una generica cella c, all'istante t, è data da:

$$\xi_t^c = \rho \cdot \tau_c^t \tag{4.5}$$

Dove τ_c^t è la quantità totale di feromone depositata sulla cella c all'istante t.

Per il calcolo di ρ viene utilizzato il coefficiente $ERTU_{\%}$ (Evaporation Rate Time Unit), che riguarda il tasso di evaporazione per unità di tempo. Siano t_v l'istante corrispondente all'ultima visita in una cella e t l'istante corrente. $(t - t_v)$ è dunque il tempo trascorso dall'ultima visita. La percentuale di feromone evaporato nell'intervallo di tempo preso in esame è dato da:

$$\rho = (t - t_v) \cdot ERTU_{\%} \tag{4.6}$$

Considerando l'evaporazione del feromone e la diffusione in accordo con la distanza dalla cella di rilascio, la quantità totale di feromone in una generica cella c, all'iterazione t, è data dall'espressione:

$$\tau_c^t = \tau_c^{t-1} - \xi_c^t + \sum_{k=1}^{N^D} \Delta\tau_c^{k,t} \tag{4.7}$$

Dove N^D è il numero totale di droni che hanno depositato il feromone.

Nella figura 4.27 è possibile osservare un esempio di diffusione feromonica.

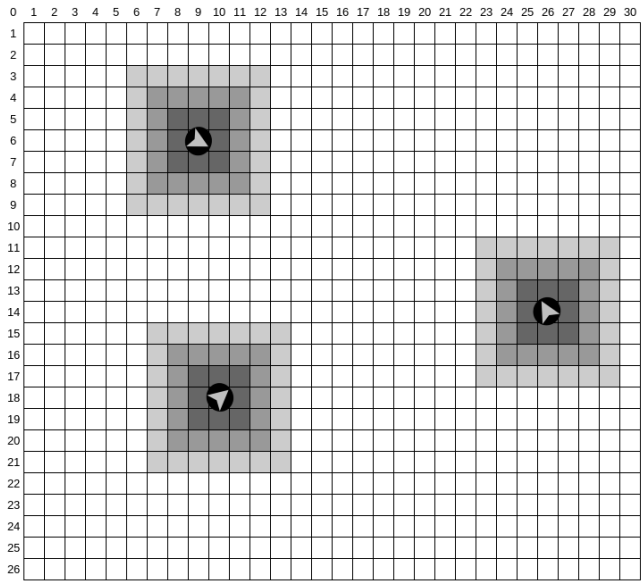

Figura 4.27: Esempio di diffusione del feromone.

Selezione cella successiva: ciascun drone k, ad ogni iterazione t, è posizionato in una particolare patch c_k^t, circondata dal set $N(c_k^t)$ di celle accessibili da quella posizione, denominato *neighborhood*.

Ogni drone rileva il feromone depositato in $N(c_k^t)$ e, successivamente, seleziona la cella in cui muoversi all'iterazione successiva.

La probabilità, all'istante t, che un agente k ha di muoversi dalla cella c_k^t ad una $c \in N(c_k^t)$ può essere calcolata attraverso la seguente relazione:

$$p(c|c_k^t) = \frac{(\tau_c^t)^\phi \cdot (\eta_c^t)^\lambda}{\sum_{b \in N(c_k^t)} (\tau_b^t)^\phi \cdot (\eta_b^t)^\lambda} \ , \ \forall c \in N(c_k^t) \tag{4.8}$$

Dove:

- $(\tau_c^t)^\phi$ rappresenta la quantità di feromone nella cella c al tick t;
- $(\eta_c^t)^\lambda$ è una variabile euristica, necessaria ad evitare che un drone resti intrappolato in un minimo locale;

- ϕ e λ sono due costanti parametriche, utilizzate per bilanciare i pesi dati al feromone ed alla variabile euristica, rispettivamente.

Il drone k selezionerà, dunque, la patch che soddisfa la seguente condizione:

$$c = min[p(c|c_k^t)] \tag{4.9}$$

In questo modo, l'agente si dirigerà verso zone inesplorate, ottimizzando l'esplorazione dell'area ed il rilevamento dei target non ancora trovati.

Algoritmo di esplorazione: questo algoritmo è un processo iterativo. Alla prima iterazione, ogni cella ha lo stesso valore di intensità feromonica, in modo da rendere quasi casuale la selezione di una patch. Alle iterazioni successive, i droni si muovono da una cella all'altra secondo le regole espresse in 4.8 e 4.9. La traccia di feromone sulle celle visitate è aggiornata secondo 4.7, così da aumentare la probabilità che le celle con intensità feromonica nulla vengano visitate.

L'obiettivo è quello di ridurre comportamenti di sovrapposizione o ridondanza, in modo da rendere ancor più efficente il completamento della missione.

L'algoritmo termina l'esecuzione nel momento in cui il drone diventa *coordinator* o *recruited*, come vedremo più avanti, oppure se la missione è completa.

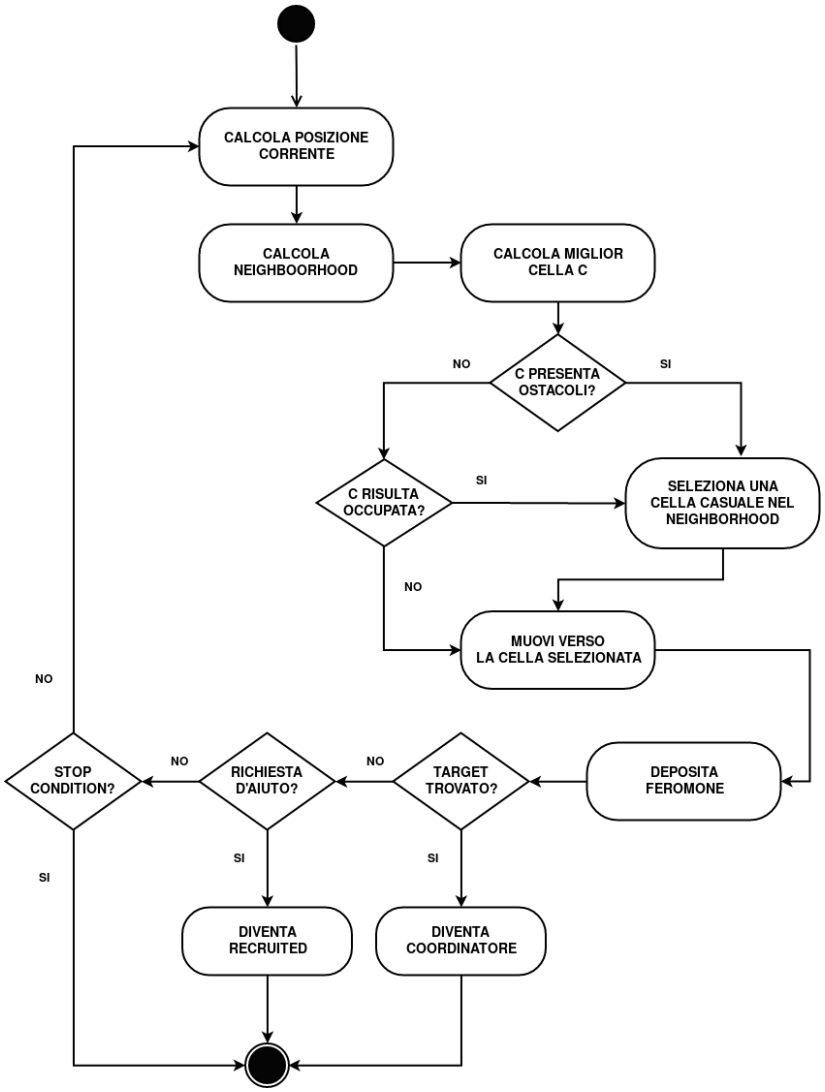

Figura 4.28: Diagramma di attività per la strategia di esplorazione.

4.5.2.2 Strategie di reclutamento

Quando un drone identifica un target, esso diventa *coordinator* ed inizia una procedura di reclutamento di altri agenti, per poter lavorare l'obiettivo in modo cooperativo. Tale procedura prevede l'utilizzo di una comunicazione wireless per rintracciare i

droni che risultano posizionati entro un raggio R_t. I pacchetti inviati contengono le coordinate del mittente che, non appena lette da un drone in stato d'esplorazione, portano quest'ultimo a cambiare il proprio stato in *recruited* ed a raggiungere il coordinatore.

Come accennato nella sezione 4.3, è necessario che un numero minimo di droni raggiunga il target, prima che questi possa essere lavorato e che la comunicazione del coordinatore venga interrotta.

Di seguito illustriamo tre differenti metaeuristiche per le gestione della fase di reclutamento.

Strategia di reclutamento FTS-RR: l'algoritmo *Firefly-based team strategy for robots recruitment* è una strategia di ispirazione biologica, basato sul comportamento delle lucciole.

In particolare, quando un drone identifica un target, questo diventa *coordinatore* ed inizia ad attrarre gli altri droni come una lucciola.

Gli elementi più importanti sono la *variazione di intensità luminosa* ed il cosiddetto *coefficiente di attrattività*. Il primo fattore dipende dalla distanza tra due lucciole, il secondo è funzione del primo e, di conseguenza, anche della distanza.

La distanza tra due lucciole i e j, posizionate rispettivamente in (x_i, y_i) e (x_j, y_j), può essere definita come la distanza Euclidea, data da:

$$r_{ij} = \sqrt{(x_i - x_j)^2 \ + \ (y_i - y_j)^2} \tag{4.10}$$

Poiché la funzione di attrattività di una lucciola varia in base a r_{ij}, si può selezionare una qualunque funzione monotona decrescente dipendente dalla distanza tra le due lucciole. Possiamo dunque utilizzare la seguente funzione esponenziale:

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{4.11}$$

Dove:

- β_0 rappresenta l'attrattività iniziale a distanza $r_{ij} = 0$;
- γ è il coefficiente di assorbimento alla fonte che regola la diminuzione di intensità luminosa.

Il movimento di una lucciola i, che è attratta da una lucciola più luminosa j, nella sequenza di tick $t \to t+1$, è regolato dalle seguenti relazioni:

$$\begin{cases} x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha(\sigma - \frac{1}{2}) \\ y_i^{t+1} = y_i^t + \beta_0 e^{-\gamma r_{ij}^2}(y_j^t - y_i^t) + \alpha(\sigma - \frac{1}{2}) \end{cases} \tag{4.12}$$

Dove:

- x_i^t e y_i^t rappresentano le coordinate della posizione corrente;
- $\beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t)$ e $\beta_0 e^{-\gamma r_{ij}^2}(y_j^t - y_i^t)$ sono utilizzati per modellare l'attrattiva della lucciola i come l'intensità della luce vista dalle lucciole adiacenti;
- $\alpha(\sigma - \frac{1}{2})$ è un termine di randomizzazione, con α parametro di randomizzazione che dipende dalla missione;
- σ rappresenta il fattore che regola la distanza di visibilità ed in molti casi può essere settato a 1.

Prendiamo adesso in esame il caso in cui un drone riceve più di una richiesta d'aiuto: in questo caso si muoverà verso il target più luminoso, secondo la relazione 4.11. Il movimento di un drone è dunque condizionato sia dalla distanza con il target che da una componente randomica, atta ad evitare che l'agente rimanga incastrato in un minimo locale.

Affinché le relazioni in 4.12 possano essere applicabili ad un contesto discreto, come nel caso dell'ambiente di simulazione, gli autori hanno trovato necessario modificarle come segue. Per la coordinata x:

$$\begin{cases} x_i^{t+1} = x_i^t + 1, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha(\sigma - \frac{1}{2}) > 0] \\ x_i^{t+1} = x_i^t - 1, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha(\sigma - \frac{1}{2}) < 0] \\ x_i^{t+1} = x_i^t, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha(\sigma - \frac{1}{2}) = 0] \end{cases} \tag{4.13}$$

Mentre per la coordinata y:

$$\begin{cases} y_i^{t+1} = y_i^t + 1, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(y_j^t - y_i^t) + \alpha(\sigma - \frac{1}{2}) > 0] \\ y_i^{t+1} = y_i^t - 1, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(y_j^t - y_i^t) + \alpha(\sigma - \frac{1}{2}) < 0] \\ y_i^{t+1} = y_i^t, \text{se}[\beta_0 e^{-\gamma r_{ij}^2}(y_j^t - y_i^t) + \alpha(\sigma - \frac{1}{2}) = 0] \end{cases} \tag{4.14}$$

Per evitare che un drone si alterni da un coordinatore all'altro, data la presenza di parametri di randomizzazione, la selezione del coordinatore viene effettuata solo nei seguenti casi:

- arriva una richiesta da un drone appena diventato coordinatore il cui raggio di reclutamento ricade sull'agente;
- il robot passa dallo stato di esploratore a quello di reclutamento;
- il robot esce dal raggio di reclutamento del coordinatore scelto, annullando così la selezione.

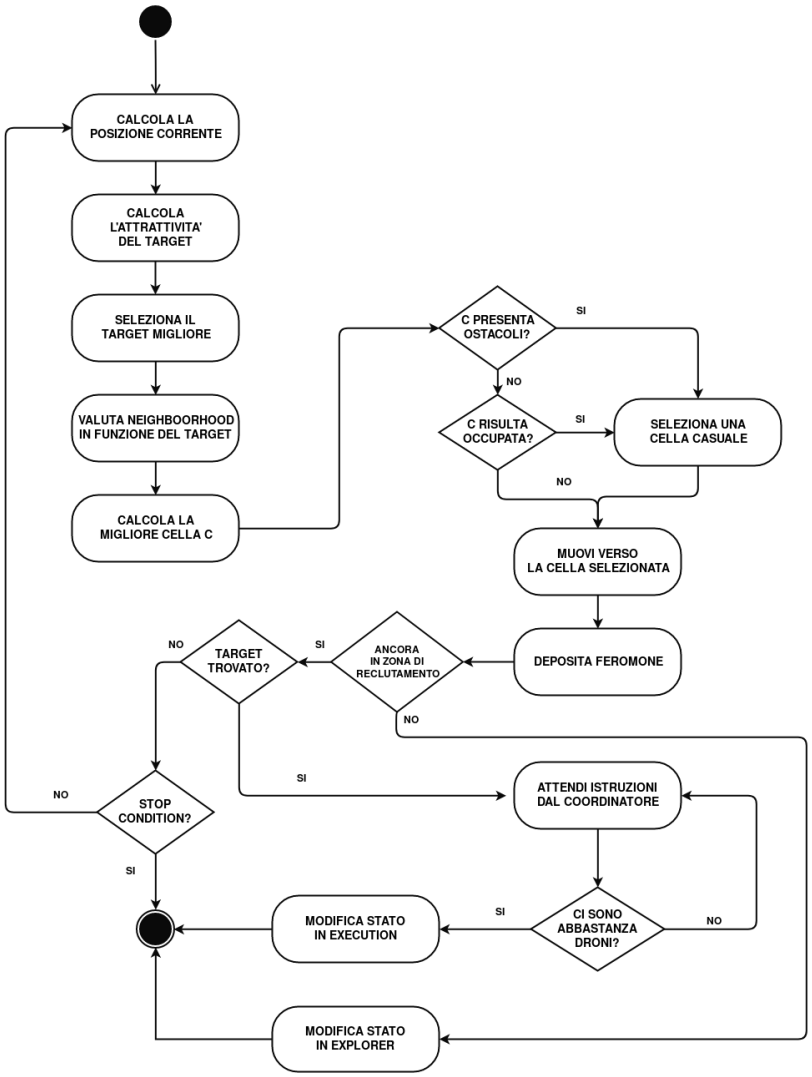

Figura 4.29: Diagramma di attività per la strategia FTS-RR.

Strategia di reclutamento PSO-RR: *Particle Swarm Optimization* è una tecnica di ottimizzazione ispirata ai movimenti degli stormi. Ogni agente i si muove nello spazio di ricerca e presenta una velocità v_i^t e delle coordinate (x_i^t, y_i^t). La velocità è aggiornata secondo la relazione seguente, in funzione delle coordinate di un target

ricevute da un drone coordinatore:

$$\begin{cases} v_{xi}^{t+1} = \omega v_{xi}^{t} + r \cdot c(x_z - x_i^t) \\ v_{yi}^{t+1} = \omega v_{yi}^{t} + r \cdot c(y_z - y_i^t) \end{cases} \tag{4.15}$$

Dove:

- ω è il coefficiente inerziale;
- c è il coefficiente di accelerazione;
- $r \in [0, 1]$ è un numero generato casualmente, per evitare blocchi in minimi locali.

Un drone i che si trova nella cella di coordinate (x_i^t, y_i^t), all'iterazione t, che riceve una richiesta d'aiuto da un coordinatore z, sceglierà la patch in cui muoversi secondo le seguenti espressioni:

$$\begin{cases} x_i^{t+1} = x_i^t + 1, \text{se}[v_{xi}^{t+1} > 0] \\ x_i^{t+1} = x_i^t - 1, \text{se}[v_{xi}^{t+1} < 0] \\ x_i^{t+1} = x_i^t, \text{se}[v_{xi}^{t+1} = 0] \end{cases} \tag{4.16}$$

e

$$\begin{cases} y_i^{t+1} = y_i^t + 1, \text{se}[v_{yi}^{t+1} > 0] \\ y_i^{t+1} = y_i^t - 1, \text{se}[v_{yi}^{t+1} < 0] \\ y_i^{t+1} = y_i^t, \text{se}[v_{yi}^{t+1} = 0] \end{cases} \tag{4.17}$$

Se il drone riceve richieste da più coordinatori, questo sceglierà di muoversi verso quello più vicino.

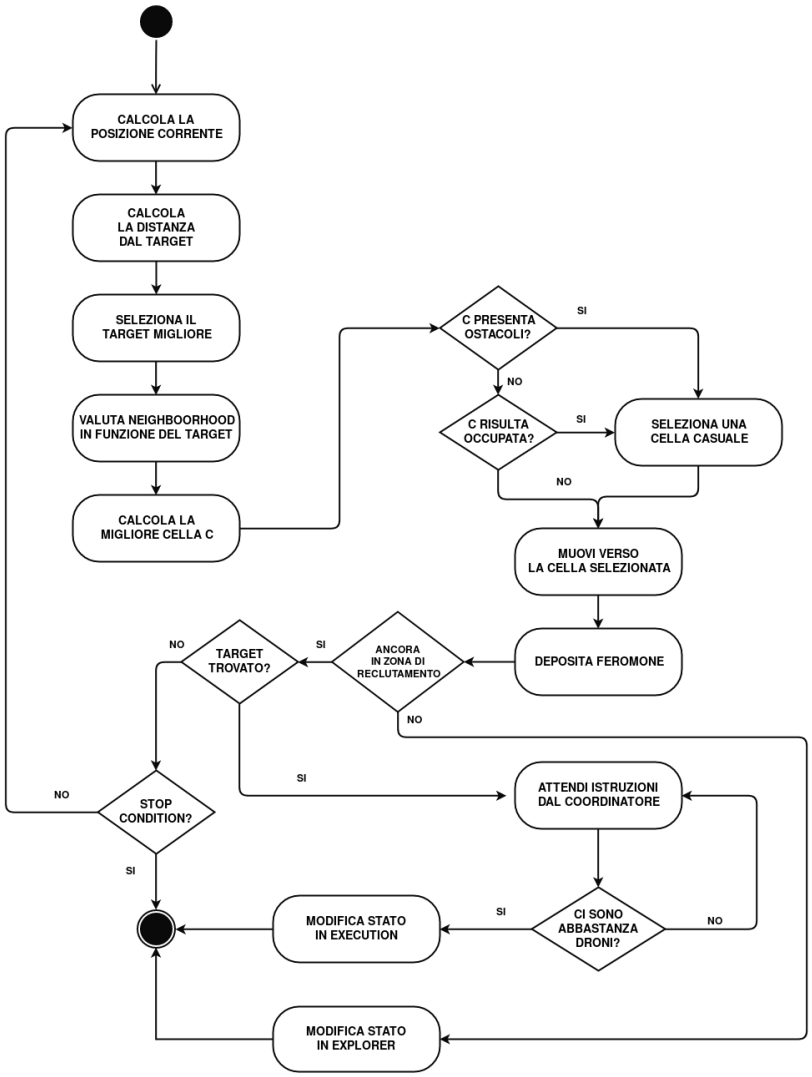

Figura 4.30: Diagramma di attività per la strategia PSO-RR.

Strategia di reclutamento ABC-RR: *Artificial bee colony for robots recruitment* è una strategia che segue il comportamento delle api in cerca di una fonte di nutrimento di qualità. Nell'algoritmo in questione, la distribuzione di fonti di cibo, all'interno del territorio di esplorazione, può essere modificata dalle api stesse nel corso del tempo. L'algoritmo utilizza l'insieme delle api per trovare la soluzione ottima al problema.

Nell'ambiente simulato, le fonti di cibo sono rappresentate dai target e la metaeuristica è applicata grazie alla comunicazione avviata dal coordinatore.

Sia $\phi \in [-1, 1]$ un coefficiente scelto in modo casuale, ogni robot i nel raggio di reclutamento di un coordinatore z, all'iterazione t, si muoverà in funzione delle seguenti equazioni:

$$\begin{cases} x_i^{t+1} = x_i^t + \phi(x_i^t - x_z) \\ y_i^{t+1} = y_i^t + \phi(y_i^t - y_z) \end{cases} \tag{4.18}$$

Se vogliamo applicare tali regole al piano bidimensionale dell'ambiente di simuazione, esse vanno adattate come segue:

$$\begin{cases} x_i^{t+1} = x_i^t + 1, \text{se}[\phi(x_i^t - x_z) > 0] \\ x_i^{t+1} = x_i^t - 1, \text{se}[\phi(x_i^t - x_z) < 0] \\ x_i^{t+1} = x_i^t, \text{se}[\phi(x_i^t - x_z) = 0] \end{cases} \tag{4.19}$$

e

$$\begin{cases} y_i^{t+1} = y_i^t + 1, \text{se}[\phi(y_i^t - y_z) > 0] \\ y_i^{t+1} = y_i^t - 1, \text{se}[\phi(y_i^t - y_z) < 0] \\ y_i^{t+1} = y_i^t, \text{se}[\phi(y_i^t - y_z) = 0] \end{cases} \tag{4.20}$$

Nel caso in cui un drone i riceva più di una richiesta di aiuto, esso dovrà decidere quale coordinatore prendere come riferimento. La *qualità* di un coordinatore z è definita come il reciproco della distanza:

$$\mu_z^i = \frac{1}{r_{iz}} \tag{4.21}$$

Dove r_{iz} rappresenta la distanza euclidea tra i e z.

La probabilità che il drone i selezioni z come coordinatore di riferimento è data da:

$$p_z^i = Pcoordinator_i = z = \frac{\mu_z^i}{\sum_{b \in RR_i} \mu_b^i} \tag{4.22}$$

Dove RR_i è l'insieme di richieste di reclutamento ricevute da i. In questo modo $\sum_{b \in RR_i} p_b^i = 1$.

Viene pertanto utilizzata la regola della *spinning-wheel*: ad ogni coordinatore è associata una probabilità. Se immaginiamo una *ruota della fortuna*, possiamo rappresentare tale probabilità come una porzione più o meno grande di questa ruota e

la dimensione della porzione dipende proprio dalla probabilità calcolata. Il drone farà girare questa ruota, in modo da ottenere il target (ed il coordinatore) di riferimento.

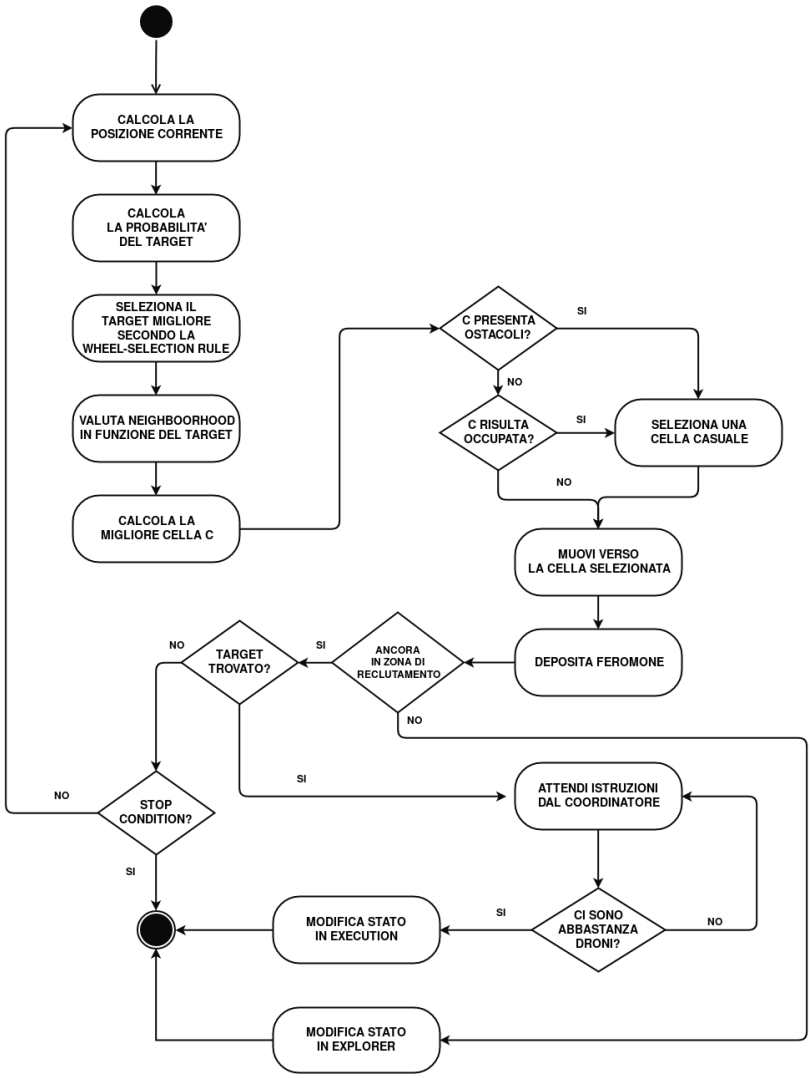

Figura 4.31: Diagramma di attività per la strategia ABC-RR.

Capitolo 5

Ottimizzazione parametrica e valutazione delle performance

L'obiettivo dell'ambiente di simulazione è quello di valutare le performance dell'algoritmo integrato per stabilire la migliore configurazione dei parametri in un dato contesto applicativo. Data l'eterogeneità delle missioni, infatti, non è possibile stabilire una configurazione di parametri che sia ottimale per tutti gli scenari.

Come abbiamo già più volte sottolineato nei capitoli precedenti, l'obiettivo di uno sciame è quello di identificare, entro il tempo di autonomia, i target presenti nell'area di esplorazione. Una valutazione delle performance, di conseguenza, può essere eseguita in considerazione del tempo impiegato per il rilevamento (o per l'esecuzione, se la missione lo richiede) di almeno il 95% dei target (*discovering time*). Una migliore configurazione dei parametri in ingresso, in relazione alla strategia di coordinamento adottata, comporterà un *discovering time* inferiore.

Un altro parametro per la valutazione delle performance è rappresentato dal numero di collisioni, che risulta importante per due ragioni fondamentali:

1. *Costi:* in un contesto reale, la perdita di un drone può avere un impatto economico piò o meno rilevante a seconda del tipo di UAV;
2. *Rischi:* la collisioni di un drone può avere ripercussioni anche sull'incolumità di persone o cose.

Al fine di ottenere un risultato significativo da un punto di vista statistico, la valutazione delle performance di un insieme di parametri viene valutata come media su più esecuzioni indipendenti del simulatore.

5.1 Ottimizzazione basata su *Differential Evolution*

L'algoritmo *Differential Evolution* è uno strumento software in grado di trovare la configurazione ottimale dei parametri per un problema caratterizzato da una specifica complessità. Risulta importante sottolineare che l'algoritmo *Differential Evolution* non garantisce la soluzione ottima del problema, ma una soluzione ottimale, ovvero una soluzione "vicina" a quella ottima.

Nel caso in oggetto, lo strumento *Differential Evolution* è utilizzato per configurare i parametri relativi al coordinamento previsto dall'algoritmo integrato nel simulatore. L'obiettivo principale del simulatore, infatti, è quello di valutare l'impatto dei meccanismi di coordinamento di uno sciame di UAV sui diversi scenari applicativi.

L'algoritmo *DE* ricerca il valore ottimale per ciascun parametro di configurazione all'interno di un intervallo definito a priori. Gli estremi di tale intervallo devono essere scelti in modo accurato in relazione alle caratteristiche del contesto applicativo e alle specifiche reali dei droni e dei sensori di rilevamento.

5.1.1 Funzionamento dell'algoritmo *Differential Evolution*

Il Differential Evolution è un algoritmo evolutivo capace di gestire correttamente funzioni non differenziabili e non lineari. La ricerca del valore ottimo inizia con una popolazione generata in modo casuale, dove i valori assunti da ciascun elemento sono vincolati ad intervalli scelti a priori. Ad ogni iterazione, viene calcolato il risultato di una funzione obiettivo, detta *fitness function*, sui vettori di parametri della generazione corrente. L'obiettivo è quello di minimizzare il risultato di questa funzione operando con i meccanismi di evoluzione differenziale.

Come molti algoritmi che rientrano in questa categoria, opera con tre operazioni fondamentali: mutazione, crossover, selezione. Lo spazio delle soluzioni viene esplora-

to attraverso dei vettori di parametri generati per differenza; questo comportamento si individua nell'operazione di mutazione. Questa rappresenta la maggior differenza con gli algoritmi genetici, che utilizzano l'operazione di crossover (rimescolamento di geni) come primo meccanismo di ricerca.

Il Differential Evolution utilizza delle differenze pesate fra i vettori delle soluzioni al fine di perturbare la popolazione e non richiede, inoltre, la codifica binaria dei membri della popolazione.

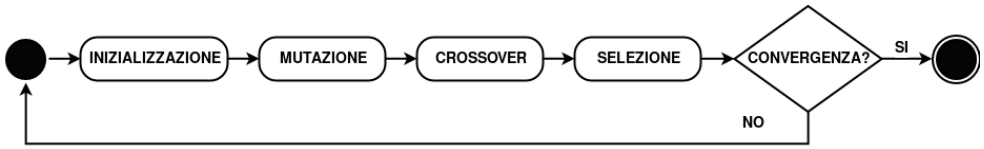

Figura 5.1: Algoritmo Differential Evolution step-by-step.

Di seguito, si riporta lo pseudocodice del *Differential Evolution*. Nei paragrafi successivi, invece, verranno descritte nel dettaglio le tre operazioni fondamentali sopra introdotte.

```
begin
    generate randomly an initial population of solutions ;
    compute the fitness function of the initial population ;
    while stop condition !satisfied do
        for each parent do
            select three solutions at random ;
            create one offspring using the DE operators ;
        end
        for each member of the next generation do
            if offspring(x) is more fit than parent(x) then
                parent(x) is replaced ;
            end
        end
    end
end
```

Figura 5.2: Pseudocodice Differential Evolution.

5.1.1.1 Mutazione

Il meccanismo di mutazione è utilizzato per produrre una popolazione di NP vettori, dove NP rappresenta il numero di individui nella popolazione. In particolare, questa operazione genera un nuovo vettore mutante, aggiungendo una frazione della differenza tra due vettori, selezionati casualmente, ad un terzo.

L'implementazione elementare di questa operazione è descritta dall'espressione di seguito riportata:

$$v_i = x_{r_0} \ + \ F(x_{r_1} - x_{r_2})$$

Dove:

- v_i è il nuovo vettore mutante;
- I termini x_{r_i}, con $i = 0, 1, 2$, rappresentano i tre vettori selezionati casualmente tra quelli disponibili;
- La funzione $F \in (0, 1+)$ regola il tasso di evoluzione della popolazione.

5.1.1.2 Crossover

Questa operazione produce i vettori da testare a partire dai vettori di parametri. In particolare, viene incrociato ogni vettore della popolazione attuale con un vettore mutante prodotto dall'operazione precedente.

L'espressione per descrivere la fase di crossover è la seguente:

$$u_i = \begin{cases} v_{i,j} \ , \text{se } rand_j(0,1) \leq Cr \\ u_{i,j} \ , \text{altrimenti} \end{cases}$$

In altre parole, il nuovo individuo viene generato scegliendo tra il vettore mutante e quello corrente, con una probabilità Cr, detta *probabilità di crossover*. Tale probabilità è un parametro definito dall'utente.

5.1.1.3 Selezione

Se il vettore della soluzione in uso, u_i, ottiene un valore della funzione obiettivo minore o uguale rispetto alla soluzione candidata migliore, questa viene sostituita

dal vettore u_i nella generazione successiva.

$$x_{i,g+1} = \begin{cases} u_{i,g} \text{ , se } f(u_{i,g}) \leq f(x_{i,g}) \\ x_{i,g} \text{ , altrimenti} \end{cases}$$

5.1.2 Implementazione software *Differential Evolution*

Al fine di ottimizzare i parametri di simulazione, è stato implementato un modulo software di evoluzione differenziale, chiamato *Differential_evolution_bridge*. Attraverso tale modulo, è possibile calcolare la funzione obiettivo tramite l'esecuzione di una simulazione con il vettore di parametri passato in input alla funzione.

La *fitness function* è calcolata come segue:

$$fitness(x) = \#ticks[simulation(x)]$$

Dove:

- x rappresenta l'individuo della generazione attuale, ovvero il vettore di parametri dato in input alla simulazione;
- $\#ticks$ rappresenta il numero di iterazioni necessarie al rilevamento del 95% di target;
- $simulation(x)$ indica l'esecuzione di una sessione simulativa con i parametri di x.

Differential evolution bridge utilizza, come implementazione software dell'evoluzione differenziale, la procedura presente nella libreria Python Scipy.Optimize, denominata *differential_evolution*. Il modello algoritmico seguito è quello offerto da Kenneth Price e Rainer Storn in [27], implementato secondo la libreria Scipy [28].

Capitolo 6

Implementazione e prove sperimentali

6.1 Simulation testbed

6.2 Sezione sui test

Ringraziamenti

Bibliografia

[1] K. Whitehead, C. H. Hugenholtz, S. Myshak, O. Brown, A. LeClair, A. Tamminga, T. E. Barchyn, B. Moorman, and B. Eaton, "Remote sensing of the environment with small unmanned aircraft systems (uass), part 2: Scientific and commercial applications," *Journal of unmanned vehicle systems*, vol. 2, no. 3, pp. 86–102, 2014.

[2] L. F. Bertuccelli and J. How, "Robust uav search for environments with imprecise probability maps," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 5680–5685, IEEE, 2005.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[4] J. A. Sauter, R. Matthews, H. Van Dyke Parunak, and S. A. Brueckner, "Performance of digital pheromones for swarming vehicle control," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 903–910, ACM, 2005.

[5] H. V. Parunak, M. Purcell, and R. O'Connell, "Digital pheromones for autonomous coordination of swarming uav's," in *1st UAV Conference*, p. 3446, 2002.

[6] C. W. Reynolds, *Flocks, herds and schools: A distributed behavioral model*, vol. 21. ACM, 1987.

[7] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Robotics and Autonomous Systems*, vol. 75, pp. 422–434, 2016.

[8] M. R. Brust, M. Zurad, L. Hentges, L. Gomes, G. Danoy, and P. Bouvry, "Target tracking optimization of uav swarms based on dual-pheromone clustering," in *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*, pp. 1–8, IEEE, 2017.

[9] C. Atten, L. Channouf, G. Danoy, and P. Bouvry, "Uav fleet mobility model with multiple pheromones for tracking moving observation targets," in *European Conference on the Applications of Evolutionary Computation*, pp. 332–347, Springer, 2016.

[10] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3866–3873, IEEE, 2014.

[11] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: communication range vs. maximum turning rate," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5015–5020, IEEE, 2011.

[12] S. A. Quintero, G. E. Collins, and J. P. Hespanha, "Flocking with fixed-wing uavs for distributed sensing: A stochastic optimal control approach," in *2013 American Control Conference*, pp. 2025–2031, IEEE, 2013.

[13] L. Bayındır, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292–321, 2016.

[14] M. Paradzik and G. İnce, "Multi-agent search strategy based on digital pheromones for uavs," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, pp. 233–236, IEEE, 2016.

[15] M. De Benedetti, F. D'Urso, G. Fortino, F. Messina, G. Pappalardo, and C. Santoro, "A fault-tolerant self-organizing flocking approach for uav aerial survey," *Journal of Network and Computer Applications*, vol. 96, pp. 14–30, 2017.

[16] H. Qiu and H. Duan, "Pigeon interaction mode switch-based uav distributed flocking control under obstacle environments," *ISA transactions*, vol. 71, pp. 93–102, 2017.

[17] N. Palmieri, X.-S. Yang, F. De Rango, and S. Marano, "Comparison of bio-inspired algorithms applied to the coordination of mobile robots considering the energy consumption," *Neural Computing and Applications*, pp. 1–24, 2017.

[18] S. Singh, S. Lu, M. M. Kokar, P. A. Kogut, and L. Martin, "Detection and classification of emergent behaviors using multi-agent simulation framework (wip)," in *Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*, p. 3, Society for Computer Simulation International, 2017.

[19] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *Journal of Artificial Intelligence Research*, vol. 53, pp. 659–697, 2015.

[20] M. G. Cimino, A. Lazzeri, and G. Vaglini, "Combining stigmergic and flocking behaviors to coordinate swarms of drones performing target search," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–6, IEEE, 2015.

[21] T. H. Labella, M. Dorigo, and J.-L. Deneubourg, "Division of labor in a group of robots inspired by ants' foraging behavior," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 1, pp. 4–25, 2006.

[22] M. G. Cimino, M. Lega, M. Monaco, and G. Vaglini, "Adaptive exploration of a uavs swarm for distributed targets detection and tracking," in *The 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pp. 1–8, 2019.

[23] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, pp. 169–178, Springer, 2009.

[24] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *arXiv preprint arXiv:1003.1409*, 2010.

[25] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4, pp. 1942–1948, Citeseer, 1995.

[26] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation*, vol. 214, no. 1, pp. 108–132, 2009.

[27] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[28] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: Open source scientific tools for {Python}," 2014.