ROBOTICA EVOLUTIVA:

Un esperimento sulla comunicazione tra robot ed espansione del tool software EvoRobot*

Facoltà di INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA
Corso di laurea in INFORMATICA

Candidato
Gabriele Carboni
1165046

Responsabile
Andrea Sterbini

Corresponsabile
Stefano Nolfi

ANNO ACCADEMICO 2012/2013

Obiettivi di questo tirocinio

L'obiettivo del tirocinio da me svolto in questi mesi al Laboratory of Autonomous Robotics and Artificial Life (LARAL) all'Istituto di Scienze e Tecniche Cognitive (ISTC) del CNR di Roma è stato quello di realizzazione di una serie di esperimenti di *robotica collettiva evolutiva* in cui un gruppo di robot viene evoluto per la capacità di svolgere dei compiti di navigazione e/o di reperimento di risorse distribuite nell'ambiente. Nell'ambito di tali esperimenti ho provveduto a confrontare il ruolo svolto da sistemi di comunicazione da me implementati.

In particolare sistemi di comunicazione "semplici", in cui ciascun robot è in grado di percepire solo la direzione di moto dei robot localizzati nelle immediate vicinanze, e sistemi di comunicazione più strutturati in cui i robot sono in grado di richiedere informazioni di tipo diverso, a seconda dei propri obiettivi correnti, e di rispondere opportunamente alle richieste ricevute. Al fine del successo degli esperimenti ho esteso il tool software di robotica evolutiva denominato **EvoRobot*** con cui sono stati progettati e simulati gli esperimenti. In particolare ho reso parallelizzabile l'algoritmo genetico evolutivo utilizzato in questo simulatore

(Algoritmo Steady State) aumentando le performance, e dando cosi la possibilità di far evolvere robot in sistemi Multi-Thread o Multi-Processore.

Indice

Capitolo 1

Introduzione alla Robotica Autonoma e Swarm Robotics

1.1 Overview

La *robotica autonoma* è la scienza che studia i metodi per progettare e realizzare *robot intelligenti* in grado di svolgere dei compiti utili, di un certo livello di difficoltà, interagendo in un ambiente fisico senza richiedere l'intervento umano.

Trae fortemente ispirazione dai *sistemi autonomi naturali* dato che sono i sistemi autonomi per antonomasia e sono in grado di svolgere in modo efficace una grande varietà di compiti in ambienti complessi e ostili. In alcuni casi infatti, lo sviluppo di robot autonomi ha l'obiettivo di costruire delle *macchine artificiali* con caratteristiche fisiche e comportamentali analoghe agli *organismi naturali*, accrescendo cosi lo studio e la comprensione dei principi alla base dell'*intelligenza biologica naturale* e dei *processi cognitivi e comportamentali degli organismi*, tra cui animali e l'uomo.

I primi risultati in robotica autonoma occorsero concretamente intorno alla metà del ventesimo secolo, con la *Machina Speculatrix* di Grey Walter nel 1953 e la *Machina Reproducatrix* di Andrew John Angyan nel 1955, in grado di esibire una serie di

comportamenti autonomi quali, esplorazione dell'ambiente sfuggendo a ostacoli, capacità di seguire un segnale di luce o sonoro, auto ricaricare le proprie batterie.

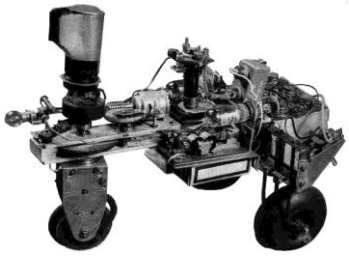

Figura 1. *Machina Speculatrix*

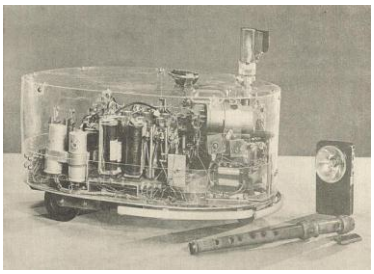

Figura 2. *Machina Reproducatrix*

Fig.1: Chiamato anche Turtle Robot, possiede un fotorecettore direzionale e un sensore di contatto i quali li consentono di riconoscere l'*intensità di luce*, esplorare l'ambiente seguendo *deboli* sorgenti di luce ed evitando *forti* sorgenti di luce e ostacoli. In particolare la Machina Speculatrix era in grado **auto-ricaricarsi all'interno di un ambiente.** [1]

Fig. 2: Molto simile alla Machina Speculatrix, fu costruita da Kretz e Zemanek e ben documentata in seguito da Although Angyan nel 1959, questa macchina era in grado di esibire capacità di discriminazione, di generalizzazione e apprendimento seguendo meccanismi di condizionamento e abituazione. [1]

Dopo un ventennio di scarso interesse sperimentale, in quel periodo concentrato maggiormente nella ricerca di sistemi d'intelligenza riproducibili all'interno di un computer ma non integrati in un robot, come la capacità di giocare a scacchi, la dimostrazione di teoremi matematici, i sistemi esperti etc., la robotica autonoma ritrovò un forte coinvolgimento nel 1986 grazie a *Rodney Allen Brooks* il quale sostenne e dimostrò un nuovo approccio per la costruzione di macchine intelligenti,

allontanandosi dai "principi base" dell'Intelligenza Artificiale del periodo, chiamato *Behavior Based Method*, inizialmente ideato e accennato da Grey Walter[1].

Attualmente, nonostante siano stati sviluppati robot molto avanzati dal punto di vista hardware, sensoriale e di potenza di calcolo, il grado di autonomia è limitato, nel senso che i robot autonomi riescono a risolvere compiti difficili ma in un ambiente specifico dedicato al contesto che si sta sperimentando, diverso da un ambiente naturale, e per un periodo abbastanza limitato.

I rapidi progressi scientifici insieme all'interesse dimostrato da diverse scienze tra cui biologiche, neurofisiologiche, psicologiche e cognitive, biomediche e genetiche, stanno portando rapidi progressi anche nel livello di autonomia.[1]

Figura 3. E-Puck

Figura 4. Darwin-OP

Figura5. iCub

Fig. 3: Robot di **piccole dimensioni** (appena 7 centimetri di diametro, e 5 di altezza con un peso di circa 200 g) integra nel suo corpo diversi sensori, tra cui 8 sensori di prossimità a infrarossi, 8 sensori di luce, una videocamera RGB 640x480, 3 microfoni e un accelerometro a 3 dimensioni. Come attuatori presenta due ruote step-motor, uno speaker, 8 led ad anello. Raggiunge una velocità massima di 13 cm/s, possiede un processore PIC da 30 MHz con una memoria RAM di 8 KB e una Flash da 144 KB, e la batteria presenta un'autonomia di circa 2 ore.
È utilizzato principalmente per esperimenti di **Swarm Robotics**, ed **Evolutionary Robotics** (tra cui l'esperimento realizzato in questo tirocinio).
Fig. 4: Piccolo **Robot-Umanoide** alto circa 27 cm in piedi e largo circa 45 cm (con braccia estese orizzontalmente), prodotto dalla ROBOTIS, possiede una grande potenza di calcolo (1.6 GHz Intel Atom Z530) ed **è utilizzato in diverse competizioni sportive** come RoboCup e FIRA di cui è stato anche vincitore nei recenti

anni. Integra una telecamera ad alta risoluzione, un giroscopio a 3 dimensioni, un accelerometro a 3 dimensioni, uno speaker e un microfono, sensori di pressione sotto i piedi e un totale di 20 giunti che permetto al robot di **assumere tante posizioni**, camminare, accovacciarsi, giocare a calcio.
Fig. 5: L'iCub è un **Robot-Umanoide** sviluppato all'**Istituto Italiano di Tecnologia** di Genova ed è adoperato in più di venti laboratori di ricerca. Ha 53 motori che permettono al robot di muovere con molta precisione testa, mani, dita, la vita e le gambe. Ha una telecamera e un microfono che gli permettono di **vedere e sentire**, sensori tattili nelle dita e nei palmi. È coinvolto in numerevoli progetti e campi di sperimentazione tra cui il **Search and Grasping**, e ITALK.

1.2 Proprietà di Embodiment e di Situatedness

In robotica autonoma, un **robot** è considerato tale se è dotato di un *corpo*, di un *sistema di controllo (o sistema nervoso)* e se interagisce in un *ambiente esterno* soggetto alle leggi della fisica e talvolta ospitante altri robot, permettendo cosi l'interazione sociale tra gli agenti.

Questi fattori, rappresentano elementi distintivi della robotica autonoma rispetto alla robotica classica e industriale e vengono identificati con i nomi di *Embodiment* e *Situatedness*.

Con il termine ***Embodiment*** si intende un robot il cui corpo contiene dei *sensori* e degli *attuatori* i quali sono essenziali per l'interazione spaziale e temporale con l'ambiente, e questi devono essere progettati ad hoc dagli sperimentatori, o sviluppati tramite processi di adattamento, in modo da facilitare lo svolgimento del compito che viene assegnato al robot, sfruttando nel modo più efficiente possibile le proprietà dell'ambiente.

Gli ***attuatori*** sono sistemi meccanici e/o elettronici che permettono al robot di muoversi all'interno dell'ambiente o di modificare l'ambiente stesso. Esempi di attuatori sono ruote, cingoli, eliche, ma anche pinze, braccia meccaniche, led, speaker etc...

I ***sensori*** sono dei dispositivi che trasformano le informazioni di natura fisica presenti nell'ambiente in informazioni interpretabili dal sistema nervoso del robot. Essi possono essere telecamere, sensori di luce, microfoni, sensori a infrarosso, sensori di calore etc. e quantificano nel tempo la variazione di una certa proprietà fisica dell'ambiente.

Tutti questi componenti forniscono al robot una *forma* e un *peso* che costituiscono fattori determinanti nella relazione *ROBOT / AMBIENTE.*

Il ***sistema nervoso*** regola come gli attuatori devono comportarsi in base allo stato dei sensori. Esso è spesso rappresentato tramite implementazioni software di Reti Neurali Artificiali in grado di simulare il parallelismo dei sistemi neurali biologici, e può essere situato sia internamente al robot, sia esternamente attraverso dei computer che si interfacciano costantemente al robot, o in alcuni casi la rete neurale può essere realizzata in hardware attraverso circuiti neurali paralleli dedicati.

Lo strato di input della rete neurale, codifica istante per istante lo *stato sensoriale di un robot*, ossia le informazioni ricevute dall'ambiente e/o dagli altri robot, mentre lo strato di output codifica le informazioni da inviare agli attuatori in modo che il

robot possa poter effettuare i movimenti idonei in funzione dello strato di input. La rete sinaptica e le proprietà di eventuali neuroni interni della rete rappresentano il cuore di un robot autonomo in quanto determinano l'output neurale che l'intero sistema nervoso produce istante per istante, regolando così come il robot deve reagire agli stimoli sensoriali ricevuti dall'ambiente o da altri agenti presenti, e quindi regola il suo comportamento.

Con il termine **Situatedness** si intende la vera e propria presenza fisica di un robot in un ambiente e la capacità di poter sfruttare le sue caratteristiche al fine di poter compiere dei task complessi in un certo intervallo di tempo.

In particolare si intende oltre che un robot posto in un ambiente, un robot in grado di sfruttare opportunamente la capacità di **modificare lo stato dei sensori attraverso l'azione**, capacità che a volte viene indicata come percezione attiva.

Questo implica principalmente due cose: A) la relazione *ROBOT* / *AMBIENTE*, in particolare la cattura delle informazioni attraverso il sistema sensoriale in un certo istante di tempo, è influenzata dalla posizione che il robot occupa nell'ambiente, B) le informazioni sensoriali che un robot può ricevere al tempo *t* dipendono dall'azione compiuta al tempo *t -1* e influenzano l'azione che dovrà compiere al tempo *t +1.*

Per apprendere le informazioni necessarie al compimento di alcuni task, è necessario quindi che il robot mostri dei comportamenti appropriati che sfruttino le proprietà dell'ambiente.

In particolare il robot capace di svolgere un task in maniera autonoma deve mostrare A) **abilità comportamentali,** come per esempio raggiungere una zona target, evitare ostacoli, afferrare un oggetto o una qualunque **azione che produce un cambiamento dell'ambiente** e/o della relazione *ROBOT* / *AMBIENTE*, utile al fine dello svolgimento del task, B) **abilità cognitive,** come per esempio riconoscere un colore, una forma di un oggetto, distinguere un predatore da una preda, ricordare la posizione di un oggetto o un qualunque **processo che produce un effetto rilevante** senza necessariamente portare modifiche all'ambiente, ma modificando esclusivamente il modo di approcciarsi ad una certa situazione sensoriale offerta dall'ambiente.[1]

Lo studio di questi comportamenti, permette di portare avanti le ricerche e gli interessi su alcuni processi comportamentali e cognitivi osservati in certe specie animali come il saper riconoscere una preda o trovare del cibo, e anche capacità sociali se si tratta di un *sistema multi-agente* come il saper **cooperare con altri agenti per svolgere compiti che un singolo agente non potrebbe mai risolvere**, oppure esibire **capacità di comunicazione per agevolare la cooperazione** di un gruppo di agenti.

1.3 Design e Sviluppo di Robot Autonomi

I robot autonomi possono essere sviluppati tipicamente seguendo (attualmente) tre diverse metodologie, le quali si distinguono per gli obiettivi posti nei relativi esperimenti, ma che tuttavia, possono essere combinate in modo da sfruttare in diverse parti di un esperimento le caratteristiche e i benefici di ognuna.

Una prima **metodologia basata su design/progettazione**, rappresenta la metodologia più utilizzata attualmente ed è caratterizzata da un approccio Top-Down in cui lo sperimentatore suddivide il problema generale in sotto-problemi più piccoli, e in seguito progetta e realizza il sistema cercando la soluzione ad ognuno dei sotto-problemi trovati.

Un esempio tipico di questa metodologia è l'approccio **Behavior-Based** ideato da Rodney Brooks che è basato sull'assunzione che il sistema nervoso debba essere progettato in modo da ottenere diversi moduli o livelli comportamentali:

in particolare i moduli devono essere organizzati, con **un'architettura di tipo gerarchico** che include moduli comportamentali di basso livello e moduli comportamentali di alto livello, i quali rispettivamente permettono ai robot di assumere capacità comportamentali di basso livello e capacità comportamentali di alto livello[1].

Un esempio semplice di architettura organizzata a livello gerarchico la si può osservare in un robot che prima di poter raggiungere una zona target in un ambiente rappresentante per esempio l'acquisizione del cibo o una stazione di ricarica, necessita di aver acquisito abilità nell'evitare gli ostacoli e nell'esplorazione visiva dell'ambiente, le quali in seguito renderanno possibile identificare una zona target da un eventuale zona di pericolo.

Una seconda metodologia chiamata **Bio-Inspired Robotics,** (o Robotica Biomimetica) guida gli sperimentatori a progettare dei robot con caratteristiche sensoriali molto avanzate e molto simili agli esseri naturali ispiranti (come pesci, salamandre, insetti, ragni) con buoni risultati in ambienti naturali.

Seguendo questo approccio sono stati realizzati diverse "robot-copie" di animali presenti in natura, in grado di camminare e/o nuotare e grazie al contributo della Computer Vision si possono realizzare robot in grado di volare evitando gli ostacoli, tenere una certa altitudine e mostrare capacità di navigazione di alto livello utilizzando le stesse capacità mostrate dagli insetti. Questa metodologia presuppone che lo sperimentatore, o il team di sperimentatori, abbia un'alta e corretta conoscenza delle caratteristiche biologiche, morfologiche e senso-motorie dell'essere naturale ispirante che si cerca di "duplicare".

Una terza metodologia utilizza **un approccio adattivo ed evolutivo** che utilizza i processi di adattamento per cambiare le proprie caratteristiche fisiche o nervose

lavorando sull'interazione con l'ambiente e sulle modifiche dell'ambiente, senza l'intervento dello sperimentatore.

In particolare l'**Evolutionary Robotics** (Robotica Evolutiva), utilizza processi di adattamento automatico basato su algoritmi evolutivi e genetici, i quali danno al robot stesso, o a più robot, la possibilità di **evolvere le proprie capacità** comportamentali e cognitive al fine di risolvere un dato problema.

Un generale processo evolutivo consiste sostanzialmente nel far **evolvere una popolazione di individui premiandoli** quando il loro comportamento esibito è utile per la risoluzione del task.

Più in dettaglio, consiste nel ripetere ciclicamente una serie di test su individui di una popolazione, i quali vengono situati in un ambiente tentando di risolvere il task assegnato, testarli singolarmente, e valutarli secondo un certo criterio scelto a priori dallo sperimentatore, per poi applicare un **processo di selezione** su tutta la popolazione, che sceglie gli individui più idonei da riprodurre per il ciclo successivo dell'evoluzione.

Questo ciclo di **TEST – VALUTAZIONE - SELEZIONE** crea un adattamento, e solo gli individui migliori proseguiranno nell'evoluzione.

La popolazione viene rappresentata come tante stringhe genetiche binarie (o di interi) in cui ogni stringa rappresenta il codice genetico di un individuo il quale,

gene per gene rappresenta anche le sue proprietà fenotipiche. Al primo ciclo del processo evolutivo le stringhe genetiche vengono inizializzate con valori casuali.

È compito dello sperimentatore definire le corrispondenze tra le caratteristiche genotipiche e quelle fenotipiche, attraverso quello che si chiama **genotype-phenotype mapping**, in particolare la corrispondenza tra i parametri dell'esperimento soggetti al processo evolutivo e i geni. É suo compito inoltre scegliere quale algoritmo genetico evolutivo utilizzare, e specificare quali sono i suoi criteri di funzionamento da considerare per il successo del processo evolutivo.

Un importante criterio è il metodo di **valutazione degli individui** il quale, oltre ad essere scelto a priori, è sempre lo stesso durante tutto l'esperimento e consiste nell'assegnare un punteggio numerico ad ogni individuo testato, secondo quella che si chiama funzione di fitness.

La **funzione di fitness** è il cuore del processo evolutivo ed è responsabile della selezione degli individui che di fronte ad un task si comportano meglio rispetto ad altri, ossia acquisiscono un punteggio numerico più alto.

Ad ogni passo generazionale i codici genetici, subiscono una mutazione in base all'algoritmo genetico e all'operatore genetico scelto per l'esperimento (mutazione, crossover, duplicazione). Questa mutazione amplia lo spazio di ricerca e aumenta cosi le probabilità che al prossimo ciclo evolutivo si sviluppi una qualunque nuova proprietà fenotipica che favorisca il successo del task.

Infine un altro fattore importante da considerare in un esperimento di robotica evolutiva è la **scelta dell'ambiente**: esso deve esprimere le proprie caratteristiche in maniera **efficiente** e **minimale**, in modo tale da facilitare l'apprendimento da parte dei robot.

Questo perché i parametri della funzione di fitness devono essere scelti considerando le caratteristiche dell'ambiente che devono essere scelte ad hoc per gli obiettivi dell'esperimento, e perché la struttura dell'ambiente e della relazione *ROBOT* / *AMBIENTE*, determinano insieme ai sistemi nervosi, sensoriali e corporei, le reazioni corporee e motorie di un robot.

1.4 Swarm Robotics o Robotica Collettiva

La S**warm-Robotics** o **Robotica Collettiva** è una sotto-area della robotica autonoma che studia lo sviluppo di sistemi robotici costituiti da diversi robot autonomi che operano insieme.

Questa area studia i metodi per far si che un alto numero di robot situati in un ambiente, con caratteristiche relativamente semplici dal punto di vista di

Embodiment, possa essere progettato per favorire lo sviluppo di alcuni comportamenti che richiedono interazione fisica tra robot e ambiente, e interazione sociale tra i diversi agenti.[2]

In particolare un *Sistema Swarm Robotics* si differenzia dagli altri S*istemi Multi-Robot* per alcuni criteri i quali caratterizzano le ricerche e gli esperimenti di questo campo:

Alto numero di robot:

Il termine swarm robotics stesso mira alla coordinazione di uno sciame (swarm). Questo implica che un sistema in cui un certo comportamento è manifestabile solo da un certo numero di robot, e non si presta ad essere portabile ad un numero superiore di robot non è considerato un *Sistema Swarm Robotics.*

Tutte le caratteristiche del robot devono essere progettate in modo da poter soddisfare criteri di portabilità a vasti sciami di robot.

Inefficienza di un singolo robot e necessità di cooperazione

È buona norma negli esperimenti dei *Sistemi Swarm Robotics* che le caratteristiche fisiche e sensoriali di un robot non gli permettano di portare a termine un task esclusivamente con le proprie capacità acquisite, ma che sia necessario che l'intero sciame di robot possa organizzarsi e coordinarsi per compiere dei task di difficoltà superiore alla portata di un singolo.

Per questo motivo lo sviluppo di un comportamento idoneo alla risoluzione di un task in un *Sistema Swarm Robotics*, deve essere progettato dando ai robot la possibilità di poter cooperare in gruppo.

Un esempio in natura rappresentante questo tipo di comportamento è mostrato dall'efficiente cooperazione delle formiche durante l'attacco di una preda molto più grande di una singola formica, la quale da sola non sarebbe mai riuscita a portare a termine l'attacco con successo.

Un altro esempio di cooperazione efficiente di questi animali è l'utilizzo dei ferormoni lasciati dalle formiche durante la ricerca di cibo: le diverse tipologia di "scie" lasciate sul terreno, permettono al resto dello sciame di capire se una certa zona dell'ambiente offre possibilità di cibarsi oppure no.

Capacità sensoriali limitate localmente

I robot utilizzati negli esperimenti in un *Sistema Swarm Robotics* devone avere capacità sensoriali ed eventualmente comunicative, strettamente locali e limitate. Questo garantisce la continua coordinazione da parte dei robot, e favorisce la cooperazione: si pensi per esempio al movimento di uno stormo di uccelli nel quale non esistono leader, e ogni singolo individuo sceglie la direzione in cui orientarsi in base ai movimenti effettuati dagli individui vicino a lui, senza considerare i

movimenti eseguiti da un individuo lontano che vola da un'altra parte dello stormo.

Quando si studiano i sistemi multi-agente come i *Sistemi Swarm Robotics,* i comportamenti mostrati dai robot durante gli esperimenti mostrano una struttura a livelli organizzata in modo gerarchico (come accennato nella prima parte di questa relazione, nella sezione dedicata al metodo Behavior-Based).

In particolare **la complessità dei comportamenti esibiti è direttamente proporzionale alla scala temporale delle interazioni tra robot e ambiente**: le interazioni *ROBOT* / *AMBIENTE* che intercorrono nell'ordine dei millesimi di secondo (scala corta) portano il robot a sviluppare dei comportamenti relativamente semplici (come per esempio l'obstacle avoidance). L'interazione nel tempo di questi comportamenti semplici a scala corta, porta lo sviluppo di comportamenti più complessi che si manifestano in scale temporali più lunghe come ad esempio le abilità di navigazione o le **abilità che richiedono un'interazione sociale** tra gli agenti dell'ambiente come ad esempio la comunicazione.

In generale i comportamenti sociali sono il frutto di un grande numero di interazioni tra gli agenti e tra ogni singolo agente e l'ambiente stesso.

1.5 Utilizzo della comunicazione in un Sistema Swarm Robotics

Per facilitare la risoluzione di un task che richiede delle abilità comportamentali di alto livello, spesso negli esperimenti di robotica collettiva viene data ai robot la possibilità di comunicare tra loro, in quanto questa importante capacità (di alto livello sociale) può essere necessaria per permettere ai robot di coordinarsi e di cooperare più efficacemente. Più in particolare viene dato ai robot la possibilità di **apprendere e di evolvere abilità comunicative**, utili alla finalizzazione del task assegnato durante un esperimento.

Nel setup di un esperimento di robotica collettiva che prevede l'utilizzo di un sistema comunicativo tra gli agenti, in particolare nella scelta dei parametri della funzione di fitness responsabile della selezione degli individui durante il processo evolutivo, lo sperimentatore deve escludere le abilità comunicative. **La funzione di fitness non deve basarsi sulle abilità comunicative** in quanto se cosi fosse, non si potrebbero studiare e notare le condizioni e i parametri che portano la nascita della comunicazione tra gli agenti, e le relazioni tra lo sviluppo dei comportamenti e lo sviluppo delle abilità comunicative.

I parametri che permetto ai robot di apprendere le abilità di comunicazione devono essere codificati in termini genetici e il loro sviluppo deve essere affidato

esclusivamente al processo evolutivo, in modo da poter scegliere la soluzione più efficiente tra le varie soluzioni offerte dallo spazio di ricerca genetico.

Studiare l'evoluzione della comunicazione tra robot inoltre offre la possibilità di capire come l'evoluzione naturale abbia influenzato l'evoluzione della comunicazione dei sistemi naturali.

Capitolo 2

Parallelizzazione di un algoritmo genetico

2.1 I fattori computazionali dei processi evolutivi

Il processo evolutivo è un processo molto costoso al punto di vista computazionale: un moderno computer da scrivania di medio costo può necessitare anche di molte settimane per poter calcolare i dati necessari al successo di un esperimento di robotica evolutiva.

Questo è dovuto principalmente da: A) *la simulazione delle interazioni tra robot e ambiente in ogni singolo test:* con particolare riferimento alla simulazione dello stato dei sensori, gli effetti dei motori, l'attivazione della rete neurale ogni istante. Questo fattore dipende dal numero di neuroni utilizzati per rappresentare la rete neurale, dal derivante numero di connessioni sinaptiche, e dal numero di robot e ostacoli presenti nell'ambiente.

B) la necessità di valutare un gran numero di individui: questo si traduce nella ricerca di un idoneo codice genetico all'interno di uno spazio di ricerca genetico il quale dipende dal determinato esperimento. Il codice genetico di un generale individuo è

rappresentato da una stringa di bit (o di interi) che codificano in termini numerici i parametri di un esperimento di robotica evolutiva che sono affidati al processo evolutivo (parametri liberi). Uno spazio genetico è rappresentato da tutti i possibili valori rappresentabili con una stringa di n bit dove n è il numero di geni, i quali codificano i parametri liberi dell'esperimento. Una mutazione del codice genetico consiste nella complementazione di un certo numero di bit scelti a caso all'interno della stringa genetica. Un genoma è valutato valido dallo sperimentatore attraverso l'osservazione delle sue proprietà fenotipiche osservabili in seguito all'esperimento, le quali si generano dopo molte centinaia di ricambi generazionali.

C) la necessità di ripetere il processo evolutivo più volte a causa del fatto che le condizioni iniziali e le mutazioni introdotte casualmente possono produrre risultati diversi in repliche diverse: all'inizio di un processo evolutivo, la popolazione genetica è inizializzata in maniera totalmente casuale, e casuale è anche la variazione (mutazione) che un codice genetico subisce ad ogni ricambio generazionale.

All'interno di un simulatore di esperimenti di robotica evolutiva, anche la posizione che i robot assumono nell'ambiente all'inizio di ogni singolo test è casuale, e in alcuni casi anche la distribuzione degli oggetti dell'ambiente è anch'essa affidata al caso. *Tutti questi fattori implicano che un singolo processo evolutivo necessita di essere replicato più volte per poter accertare i suoi risultati*, affidandosi ogni volta ad una

successione di numeri casuali diversa (ottenendo dei risultati comportamentali a volte diversi).

2.2 Parametri di parallelizzazione di un algoritmo genetico

Come accennato nell'introduzione alla robotica evolutiva, un algoritmo genetico coinvolge una popolazione di un numero fisso di individui rappresentati da un codice genetico, i quali vengono testati uno ad uno all'interno di ogni generazione.
Un singolo test di un individuo è una successione di un numero finito di step fissato a priori dallo sperimentatore che dipende dalla complessità del task in cui ogni singola rete neurale deve aggiornare il proprio stato sensoriale ed eseguire una propagazione per generare un output motorio.
Spesso per acquisire capacità comportamentali robuste è bene che i robot interagiscano con diverse varianti dell'ambiente in modo da acquisire la capacità di svolgere il compito dato in situazioni diverse. Testare un singolo individuo in diverse condizioni ambientali, più in particolare con una **diversa combinazione spaziale degli oggetti dell'ambiente** rappresenta un primo parametro su cui basare la parallelizzazione: *affidando ad ogni singola unità di calcolo disponibile il test sequenziale di un individuo in uno specifico ambiente,* e facendo una media dei dati

ricavi dai singoli test si avrà il vantaggio di avere un evoluzione più robusta ai cambiamenti dell'ambiente. Un ulteriore vantaggio di questo fattore di parallelizzazione è rappresentato dalla bassa complessità delle informazioni da fornire ad ogni singola istanza di calcolo (informazioni sull'ambiente specifico), e delle informazioni da recuperare al termine del calcolo parallelo per la valutazione degli individui (è necessario fare una semplice media dei singoli valori di fitness).

Il fine principale di un processo evolutivo è fare evolvere un intera popolazione genetica in modo tale da poter scegliere il genoma degli individui più evoluti, ed utilizzarlo per portare a termine un task con successo.

Testare le performance di un intera popolazione genetica, vuol dire codificare il genotipo di ogni individuo in un sistema nervoso di un robot o di più robot (nel caso di robot cloni), lasciare vivere il robot o i robot nell'ambiente, e assegnare un punteggio in base a quanto il comportamento esibito sia stato utile per la realizzazione del task. **Il numero degli individui di una popolazione genetica** rappresenta un secondo e importante parametro di parallelizzazione di un algoritmo genetico: *assegnando ad ogni unità di calcolo disponibile la valutazione sequenziale di un individuo della popolazione genetica* (che di solito varia tra 20/100 individui) si avrà il vantaggio di parallelizzare la ricerca degli individui migliori di

una certa generazione, e di poter frazionare il tempo necessario al test di un intera popolazione di un fattore pari al numero di unità di calcolo parallelo disponibili.
La parallelizzazione basata sul numero di individui necessita di fornire ad ogni unità di calcolo un duplicato dell'intero esperimento: ogni individuo necessita di un ambiente con tutti i suoi componenti, di un robot o di un insieme di robot su cui incorporare le caratteristiche genetiche, il sistema senso-motorio, etc. e di strutture dati capaci di valutare i migliori individui. (Le differenze tra una versione mono-thread e una versione multi-thread dell'algoritmo Steady State verranno spiegate nel capitolo 3).

Come accennato sopra il caso rappresenta un fattore molto importante nel proseguo di un processo evolutivo poiché la generazione dei numeri casuali è affidata al calcolatore, e viene generata da un seed (numero intero) settabile a priori dallo sperimentatore. La **replica di un esperimento con diversi seed casuali** può generare diverse situazioni comportamentali alcune favorevoli e altre non favorevoli al successo del task. Questo rappresenta un terzo parametro di parallelizzazione, che avvantaggia la ricerca di un genotipo valido in uno spazio genetico e aiuta a rafforzare le statistiche di un esperimento: *assegnare ad ogni singola unità di calcolo un intero processo evolutivo comporta la duplicazione di un intero*

esperimento con la sola differenza di doversi affidare ad una sequenza di numeri casuali diversa.

Una parallelizzazione sul numero di semi casuali non presenta la necessità di gestire le informazioni ricavate da ogni singola sequenza di calcolo, poiché dopo la fase di elaborazione parallela ogni processo evolutivo ha già finito la sua evoluzione e ogni singola unità di calcolo ha calcolato i propri genomi derivati dal seed casuale iniziale.

*2.3 Espansioni software e parallelizzazione del simulatore EvoRobot**

2.3.1 Introduzione al tool software

Il simulatore EvoRobot* è un ottimo strumento per creare esperimenti in ambito di robotica evolutiva. É stato ideato e sviluppato presso il LARAL (Laboratory of Autonomous Robotics and Artificial Life) all'Istituto di Scienze e Tecniche Cognitive (ISTC) del CNR di Roma e permette di effettuare esperimenti sia in simulato che con robot reali che interagiscono in ambienti fisici.

É progettato a classi ed è scritto completamente in C++, utilizzando il framework Nokia QT (ver. 4.8), il quale gli permette di avere un interfaccia molto semplice e facile da capire ed utilizzare. Al suo interno contiene classi e metodi idonei per

- generare ambienti con oggetti ed ostacoli
- creare e gestire le reti neurali rappresentanti il sistema di controllo
- creare e gestire sensori e motori
- rappresentare geneticamente gli individui e lanciare gli algoritmi genetici evolutivi
- gestire gli esperimenti e le statistiche
- creare la funzione di fitness adatta ad un esperimento
- monitorare i robot durante i test

Molte delle caratteristiche dell'esperimento che si vuole compiere, e dell'ambiente sul quale compierlo, sono pre-settabili attraverso file di configurazione esterni al programma.

Inoltre grazie diversi tool grafici è possibile controllare lo stato di avanzamento dell'evoluzione e vedere i punteggi ottenuti dai robot generazione per generazione.

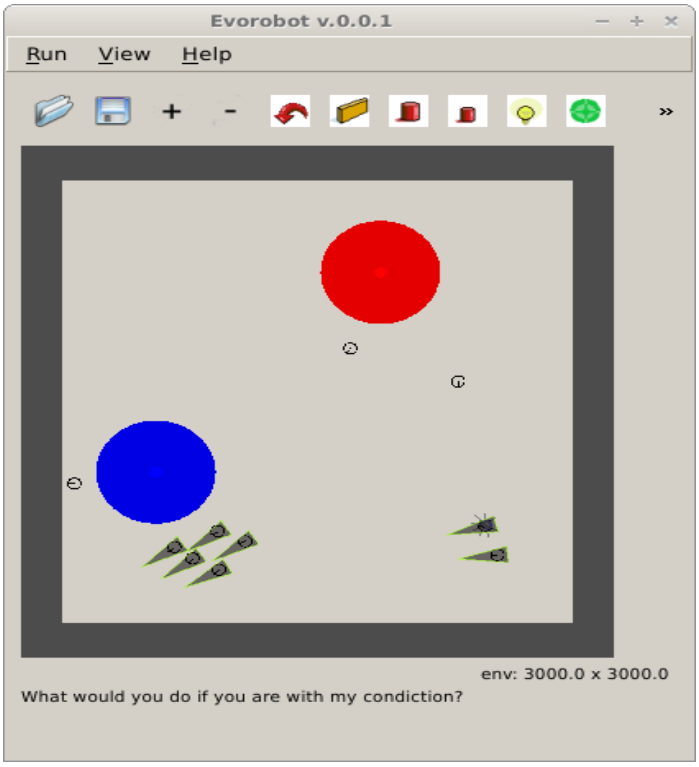

Fig. 6: Ambiente di sperimentazione. L'immagine mostra come si presenta EvoRobot* in particolare l'ambiente di sperimentazione e in alto alcuni strumenti per aggiungere oggetti nell'ambiente. Dai menù è possibile utilizzare i vari tool grafici per avere informazioni sullo stato di una evoluzione o di un test.

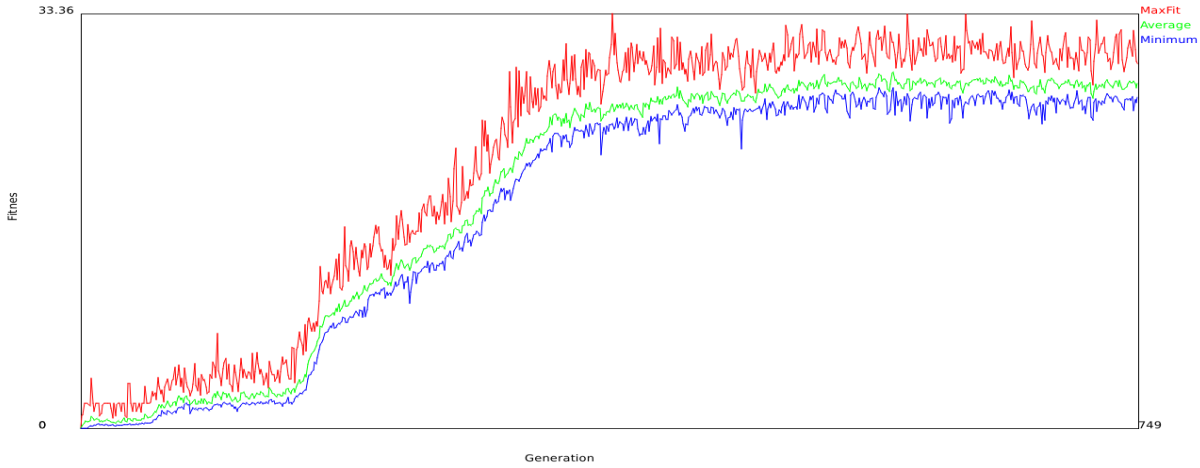

Fig. 7: Fitness monitor. Questo tool permette di osservare l'andamento di un intero processo evolutivo in particolare i punteggi ottenuti da una popolazione genetica, generazione dopo generazione. Se il Fitness Monitor si apre in fase di evoluzione, esso si aggiorna ad ogni cambio generazionale dando a disposizione dello sperimentatore i dati ottenuti finora. Si presenta con tre curve indicanti rispettivamente: il punteggio ottenuto dall'**individuo più performante in**

rosso, **la media dei punteggi** ottenuti da tutti gli individui in verde e il punteggio ottenuto dall'**individuo meno performante** in blu, in funzione del tempo (generazione per generazione).

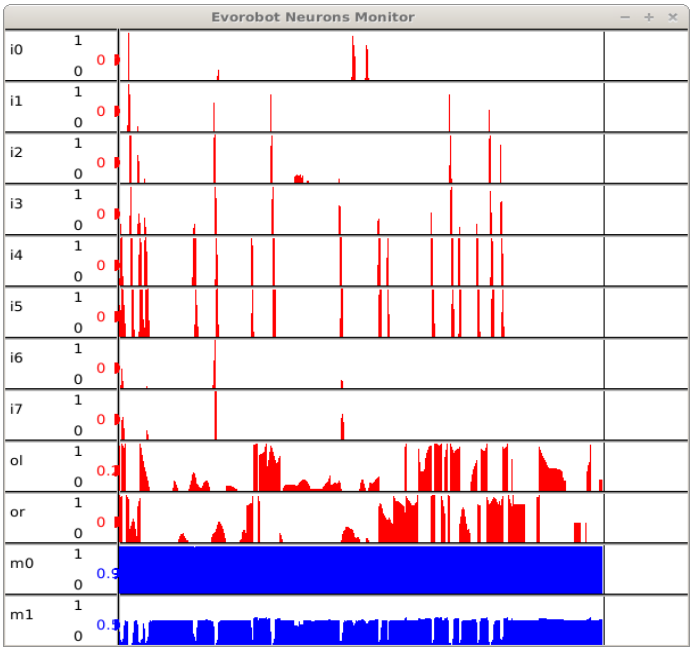

Fig. 8: Neurons Monitor. Durante la fase di test questo tool permette di vedere come reagisce il sistema senso-motorio di un robot evoluto mentre interagisce con l'ambiente. In particolare si può vedere lo stato di attivazione dei neuroni (sensoriali, interni e motori). Ogni riga nell'interfaccia rappresenta lo stato di attivazione di un singolo neurone in funzione del tempo. In rosso si rappresentano i neuroni di INPUT , in blu i neuroni di OUTPUT e in grigio, gli eventuali neuroni HIDDEN.

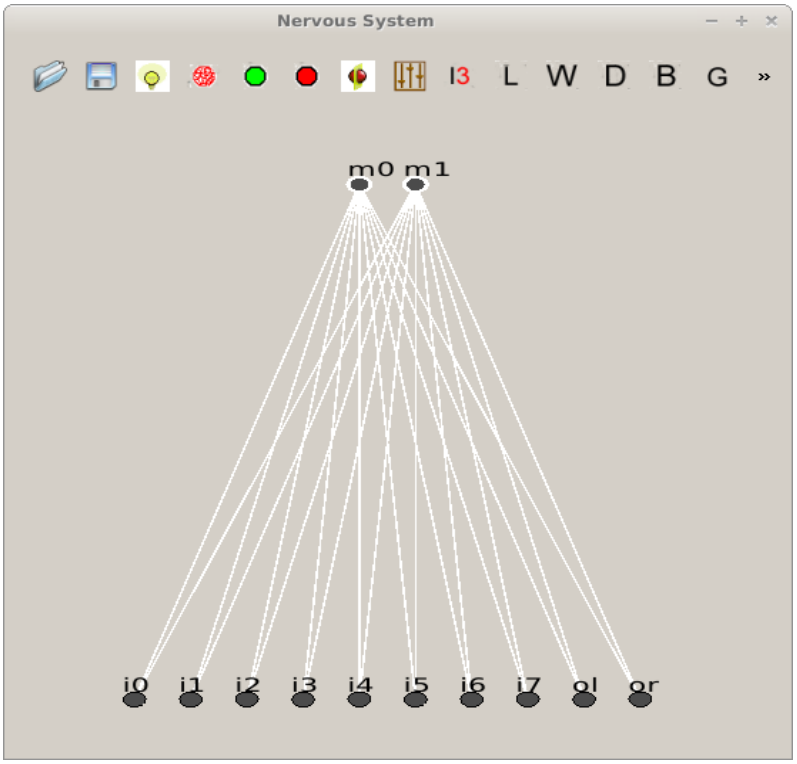

Fig. 9: Nervous System: Questo tool permette di vedere l'architettura della rete neurale rappresentante il sistema nervoso dei robot dell'esperimento. Lo strato in basso indica lo strato di input, ossia le informazioni sensoriali acquisite dall'ambiente, sia fisico che sociale. Questa figura in particolare mostra una rete neurale con tutti i pesi sinaptici uguali a 0: il colore bianco delle sinapsi che lega lo strato di input a quello di output rappresenta il valore 0 ossia una mancanza di legame tra i due strati. Quando si carica il genotipo e lo si incorpora in un robot, i pesi sinaptici si colorano in base al loro grado di eccitazione o di inibizione.

2.3.2 *Espansioni software realizzate*

I moduli principali del simulatore si possono descrivere nel diagramma seguente:

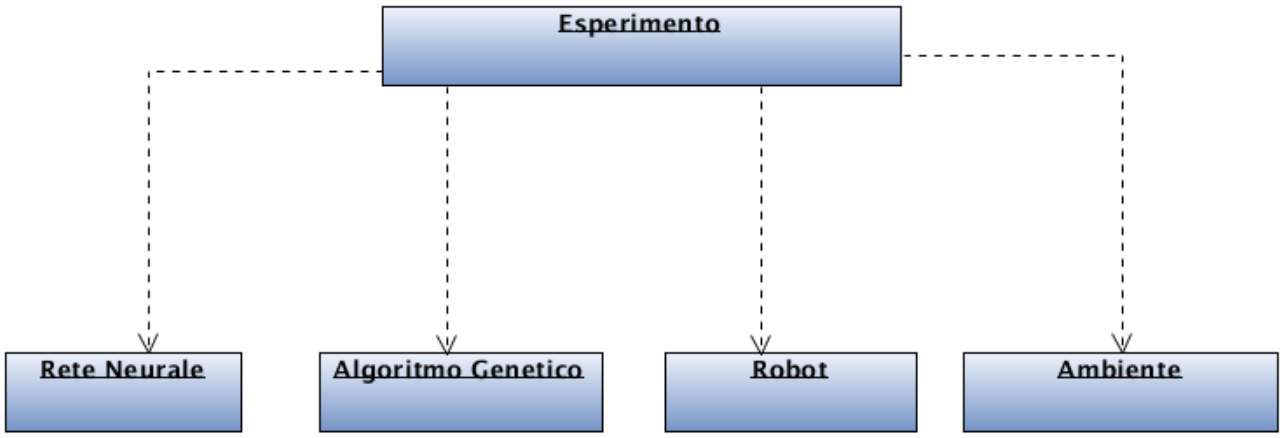

Fig. 10: Component Diagram di EvoRobot*: Il componente principale del simulatore è formato dalla classe esperimento la quale contiene un riferimento ad gli altri moduli necessari per l 'esperimento.

Nella immagine seguente invece si evidenziano le espansioni software da me realizzate nei relativi moduli, alcune delle quali sono brevemente spiegate in seguito altre nelle relative sezioni:

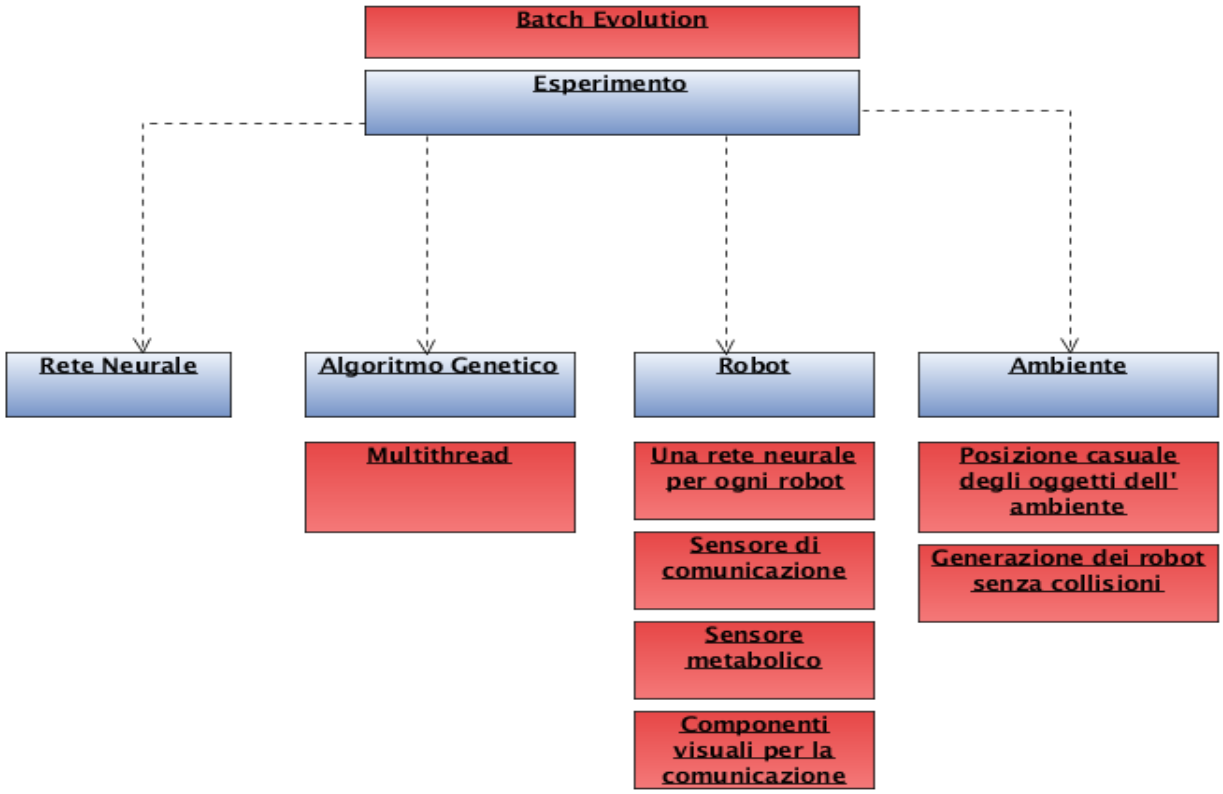

Fig. 11: Component Diagram di EvoRobot*: In rosso le espansioni software realizzate in questo tirocinio.

Implementazione di una rete neurale per ogni robot:

Prima di questo tirocinio infatti i robot condividevano una stessa istanza della rete neurale che veniva assegnata a turno a tutti robot, caricando ogni volta lo stato sensoriale di un robot e propagando la rete.

Per ragioni legate agli esperimenti è stato necessario implementare una rete neurale per ogni robot. Il che porta benefici anche in contesti esterni a questo tirocinio in quanto permette di simulare esperimenti sulla co-evoluzione e il co-adattamento nello stesso ambiente di individui con reti neurali diverse e quindi con caratteristiche genetiche e comportamentali diverse, come ad esempio il paradigma Preda - Predatore, che prima non era sperimentabile con questo simulatore.

Miglioramento del sistema di generazione di coordinate dei robot :

Durante l'inizializzazione dei test, soprattutto quando il numero di robot era abbastanza alto e l'ambiente molto piccolo, ad alcuni robot venivano assegnate delle coordinate che generavano delle collisioni tra i robot e gli oggetti dell'ambiente e/o tra i robot stessi: questo durante il test, negava completamente la mobilità ai robot coinvolti nelle collisioni e abbassava la media della fitness assegnata a tutto lo sciame.

Questa espansione è risultata necessaria per il successo degli esperimenti eseguiti in questo tirocinio.

Possibilità di generare ambienti con disposizione casuale degli oggetti:

Questa espansione consiste nel generare in modo casuale senza generare collisioni e con parametri settabili dai file di configurazione le coordinate spaziali dei componenti dell'ambiente, in ogni generazione. Permette di testare un singolo individuo in diverse condizioni ambientali, e (come accennato prima) permette ai robot di essere più attivi nell'ambiente e più reattivi alle modifiche dell'ambiente.

Visualizzazione grafica dell'attività di comunicazione tra i robot:

Questa espansione è risultata utile per visualizzare l'attività di comunicazione esercitata dai robot coinvolti nell'esperimento realizzato in questo tirocinio, illustrato nella sessione successiva.

La comunicazione è visualizzata tramite una semplice bussola che compare (o scompare) sopra i robot quando essi iniziano (o terminano) una attività di comunicazione. La bussola permette di capire il funzionamento del sistema di comunicazione, in particolare le influenze che la comunicazione ha nelle scelte di navigazione dei robot. Per implementare ciò ho utilizzato le classi e i metodi del Framework QT utilizzate per la visualizzazione dei robot e degli oggetti

dell'ambiente. Lo sperimentatore può scegliere se visualizzare o no la bussola tramite un parametro settabile nel file di configurazione dell'esperimento.

Possibilità di lanciare le evoluzioni in batch.

Questa espansione permette di poter lanciare gli esperimenti senza implementazioni grafiche risparmiando tempo computazionale necessario all'aggiornamento delle posizioni dei robot nell'ambiente, e permettendo cosi di poter eseguire un processo evolutivo nel sistema cluster del LARAL attraverso il protocollo ssh. Per realizzare ciò è stata necessaria una completa ristrutturazione della gerarchia delle classi rappresentanti i robot e l'ambiente, e i componenti grafici necessari per la loro visualizzazione.

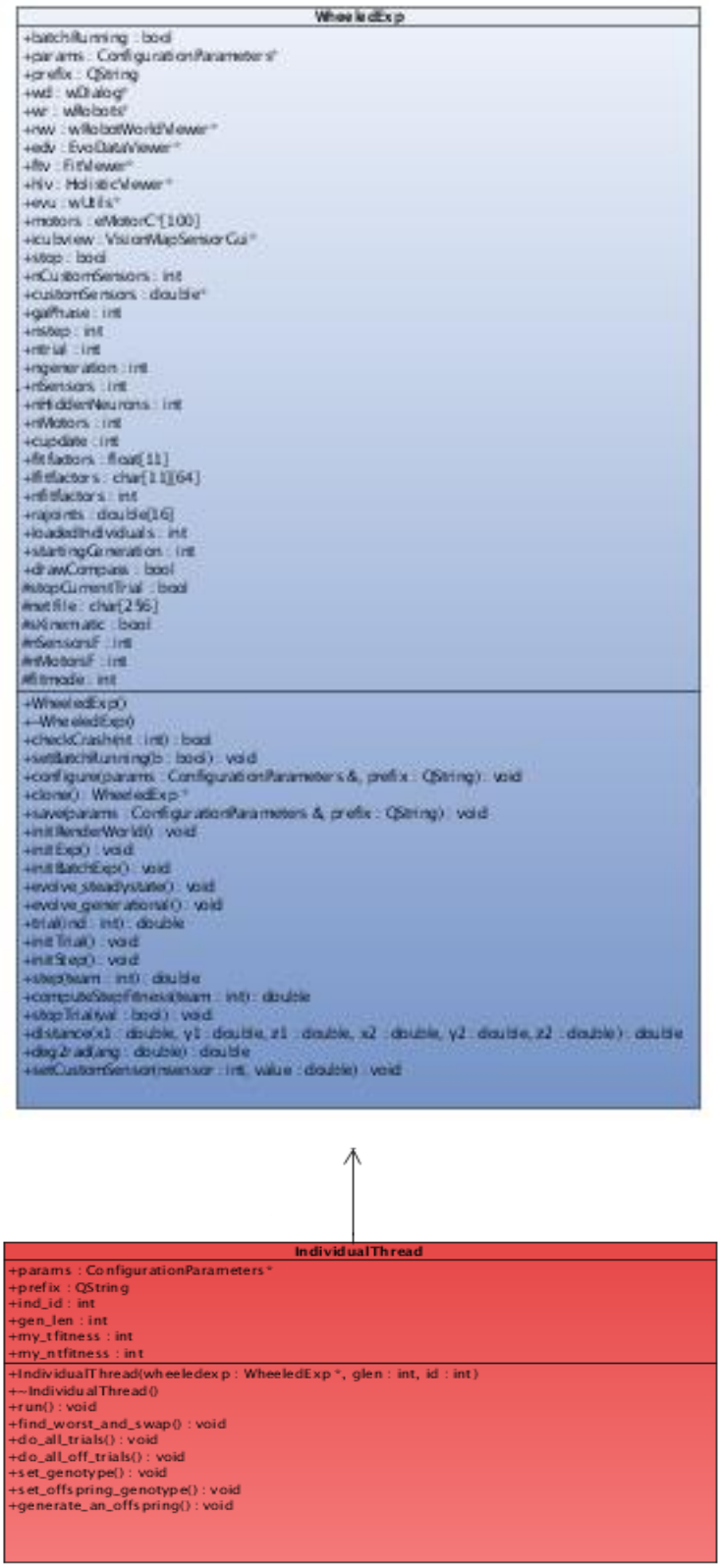

Fig. 12 Classe WheeledExp: (in blu) è responsabile di tutto l'esperimento e ha al suo interno i riferimenti a tutti i moduli necessari all'esperimento. In rosso è mostrata la sua sotto-classe IndividualThread (da me implementata e spiegata in seguito) responsabile della gestione dei singoli thread che si occupano dell'evoluzione dei singoli individui, e i metodi e le strutture dati utili a reperire le informazioni al termine del calcolo parallelo.

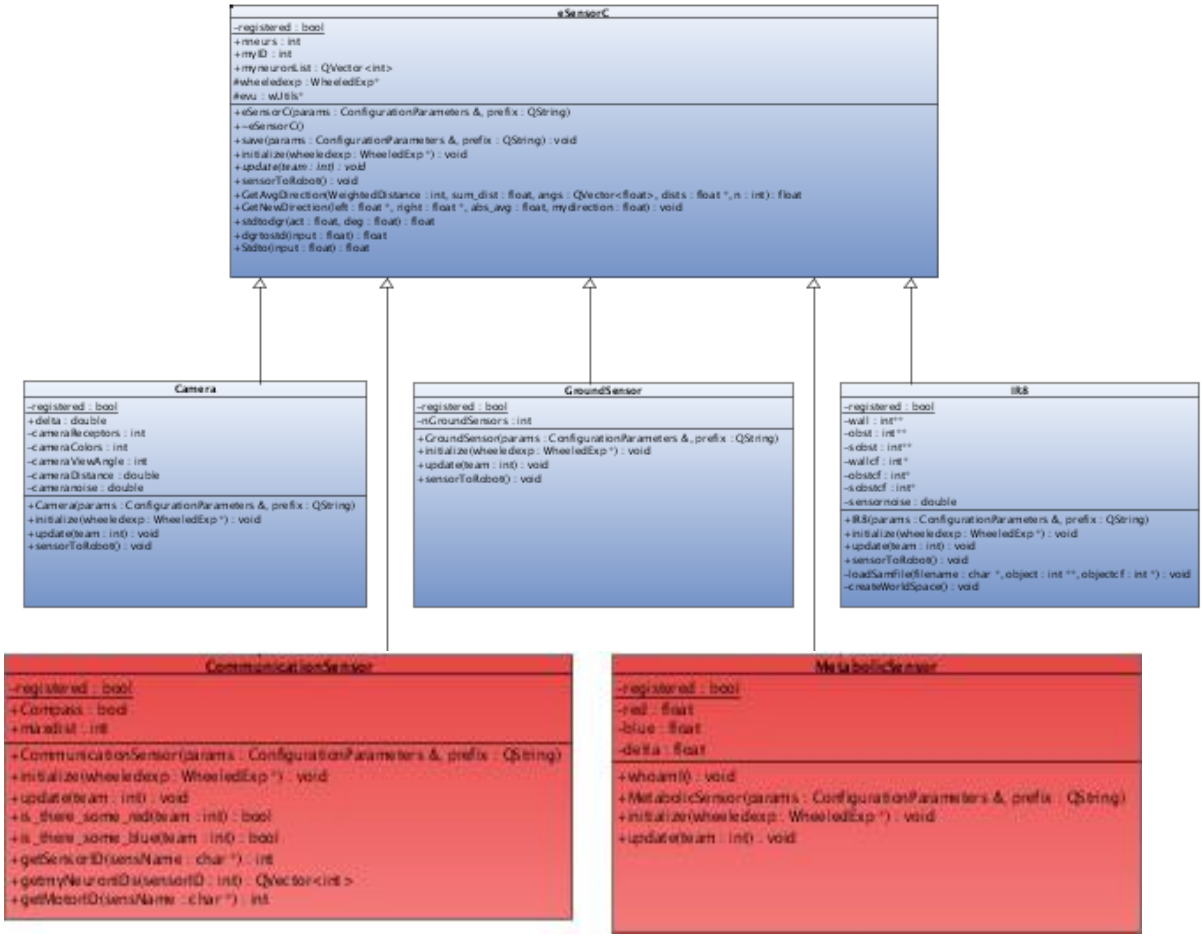

Fig. 13 Classi per la gestione dei sensori: La figura mostra la gerarchia delle classi utili alla gestione dei sensori. In blu sono rappresentate alcune classi già implementate in EvoRobot* e utilizzate per gli esperimenti da me svolti durante questo tirocinio. In rosso le classi da me implementate per la gestione dei sensori caratteristici degli esperimenti da me svolti. Il funzionamento di quest'ultimi sarà illustrato nella sessione successiva.

2.3.2 L'algoritmo Steady State (mono-thread)

Il simulatore EvoRobot* opera il suo processo evolutivo attraverso l'algoritmo genetico Steady State. Questo algoritmo si differenzia da altri algoritmi genetici poiché è un **algoritmo a riproduzione costante** (*steady-state reproduction*) che opera generando un figlio per ogni individuo della popolazione il quale viene utilizzato per rimpiazzare il genotipo peggiore della popolazione, solo se quest'ultimo ha una fitness inferiore [5]. Riporto lo pseudocodice della versione mono-thread dell'algoritmo steady state presente su EvoRobot* :

```
for each generation
.
.      for each individual ind in POPULATION
.      .
.      .      t = 0
.      .      while  (#tests < t)
.      .      .
.      .      .      fitness = test(ind)
.      .      .      ind.fitness += fitness
.      .      .      ind.time_tested ++
.      .      .      t ++
.      .       .
.      .      child  =  generate_a_variation_of( ind)
.      .      child.fitness  = 0
.      .      child.time_tested = 0
.      .      t = 0
.      .
.      .      while  ( #tests < t )
.      .      .
.      .      .      fitness = test (child )
.      .      .      child.fitness  += fitness
.      .      .      child.time_tested ++
.      .       .     .
.      .      worst = find_the_worst_individuals ( tfitness, ntfitness)
.      .
.      .      if     (      child.fitness / child.time_tested      >
.      .                    worst.fitness / worst.time_tested      )
.      .
.      .      then          write_genecode(child → worst)
```

Pseudocodice dell'algoritmo genetico evolutivo Steady State (versione mono-thread).

Questo tipo di algoritmo si allontana dalla realtà biologica in quanto assume che gli individui possano sopravvivere un numero potenzialmente indefinito di generazioni.

L'algoritmo è composto sostanzialmente da due cicli while che eseguono le fasi di test: la prima fase riguarda il codice genetico dell'individuo *ind*, mentre la seconda fase consiste nel testare una sua mutazione child ossia il codice genetico di un figlio di *ind*. Se al test t un figlio *child* si dimostra migliore del peggior individuo testato sin'ora, il peggiore individuo viene eliminato e il suo codice genetico viene sovrascritto con il codice genetico di *child*.

In questo modo ogni volta che si testa un individuo si sostituisce l'individuo peggiore fino ad ora trovato ma solo se ne esiste uno che si è comportato in maniera più performante, e si conserva il codice genetico degli individui migliori.

2.3.3 Algoritmo Steady State (multi-thread)

La parallelizzazione dell'algoritmo steady state è stata effettuata sul numero di individui.

Nella versione multi-thread implementata in questo tirocinio gli **individui sono rappresentati da una classe IndividualThread** contenente un intera istanza dell'esperimento (quindi i robot, l'ambiente, le reti neurali, etc.) ed è responsabile:

- del test di ogni individuo
- del test del figlio di ogni individuo
- della raccolta dei punteggi presi da ogni individuo e da ogni figlio

Il processo padre rappresentante l'istanza in memoria dell'esperimento è responsabile della creazione dei vari threads durante il processo evolutivo, e raccoglie gli individui (i threads) in un array "individuals". Il numero di thread da dedicare al calcolo del processo evolutivo è settabile dal file di configurazione di tutto l'esperimento e le migliori performance si avranno con un numero di unità di calcolo uguale o maggiore al numero di individui scelti per l'esperimento.

Grazie al metodo Map della classe QtConcurrent è possibile creare i threads e iniziare l'evoluzione in parallelo. Questo metodo chiede in input un array e una funzione, la quale deve essere progettata per essere lanciata parallelamente su ogni elemento di questo array. La classe QFuture rappresentante il risultato di una qualunque operazione computazionale, mantiene i risultati dei singoli individui e permette ai threads di sincronizzarsi l'un l'altro, in modo da poter utilizzare i risultati delle singole operazioni in futuro. Un suo metodo importante chiamato waitForFinished aspetta il termine delle relative operazioni parallele per proseguire nelle istruzioni relative alla valutazioni degli individui (processo padre).

Riporto uno pseudo codice della versione multi-thread dell'algoritmo stady state.

```
for each generation
.     for each individual ind in population
.     .
.     .     individuals[ind].genome            = ind.genome
.     .     individuals[ind].fitness           = ind.fitness
.     .     individuals[ind].time_tested       = ind.time_tested
.
.     future = Map( start_parallel_evolution(), individuals)
.     future.waitForFinished()
.
.
.     for each individual ind in  individuals
.     .     parents[ind].fitness    = individuals[ind].fitness /
.                                     individuals[ind].time_tested
.
.
.     for each child ch in population
.           children[ch].fitness    = individuals[ch]->cfitness /
.                                     individuals[ch].ctime_tested
.
.     Sort(parents)
.     Sort(children)
.     Merge(parents, children)
```

Pseudocodice dell'algoritmo genetico evolutivo Steady State (versione mono-thread).

```
start_parallel_evolution(individuals ind)
{
      ind.do_all_trials();

      ind.generate_an_offspring();

      ind.do_all_off_trials();
}
```

Pseudocodice della funzione start parallel evolution, caratterizzante l'algoritmo genetico evolutivo Steady State (versione multi-thread).

L'algoritmo è composto da una prima fase (sequenziale) in cui le informazioni degli individui vengono caricate (dal processo padre) nelle strutture dati dedicate al processamento di ogni singolo individuo/thread, una fase di test nell'ambiente dei singoli individui (in parallelo) in cui vengono testati padri e figli, una fase di salvataggio dei risultati, e una parte (sequenziale) di selezione degli individui

migliori tra i padri e i figli (la procedura merge ordina i padri e i figli in base al punteggio di fitness presa durante il test).

É necessario accennare un modifica di funzionamento della versione multi-thread che riguarda l'influenza del caso all'interno del ciclo evolutivo: sia nella versione mono-thread che in quella multi-thread lo sperimentatore può scegliere il numero di repliche da effettuare e il seed casuale di partenza, il quale viene incrementato replica dopo replica garantendo un seed diverso ogni volta. Questo seme genera una sequenza di numeri casuali interna al processo, i quali vengono utilizzati in ordine sequenziale durante le chiamate alle relative funzioni del processo che richiedono un numero casuale durante il processo evolutivo (es. funzione mrand() in C++, responsabili del posizionamento dei robot, i bit da mutare durante la fase di mutazione, etc.) individuo per individuo. Nella versione multi-thread lo sperimentatore sceglie il seed di partenza, questo seed genera una sequenza di numeri casuali (interna al processo padre contenente l'esperimento) e in seguito viene ricopiato l'intero esperimento per ogni individuo-thread. In questo caso *l'ordine delle chiamate che richiedono un numero casuale dipende dall'ordine in cui i thread vengono schedulati dal sistema operativo*, a differenza della versione mono-thread nella quale le chiamate avevano un ordine stabilito dall'ordine sequenziale delle istruzioni dell'algoritmo).

Questo comporta il fatto che nella versione mono-thread l'esperimento poteva anche essere ripetuto dando gli stessi risultati in quanto la sequenza dei numeri casuali era unica e associata al seed. Nella versione multi-thread invece non si può più ripetere un esperimento con gli stessi risultati, questo perché nonostante la sequenza generata sia la stessa per tutti i threads, l'ordine di accesso alla sequenza dipende dalla schedulazione dei processi e dei threads in esecuzione (compito affidato al sistema operativo), e si è deciso di non intervenire sulla ripetibilità degli gli esperimenti, ma di condividere il generatore di numeri casuali tra i diversi thread, che lo usano concorrentemente.

2.4 Dati e statistiche del calcolo multi-thread

La seguente tabella mostra alcuni dati di alcune evoluzioni eseguite mantenendo gli stessi parametri, prima con la versione mono-thread e poi con la versione multi-thread, mostrando come effettivamente il tempo di computazione di una generazione si fraziona di un fattore pari circa al numero di unità di calcolo parallele utilizzate:

	N° Thread e Speed-up teorico	Tempo medio per una generazione (min.)	Tempo per 50 generazioni (min.)	%Speed-up reale
Collective Navigation	1	0.096914	4.8457	
Collective Navigation	2	0.0513084	2.56542	94%
Collective Navigation	4	0.0289087	1.44543	83%

Capitolo 3

Un esperimento sulla comunicazione tra robot

3.1 Introduzione

In questo tirocinio ho realizzato una serie di *esperimenti di robotica collettiva evolutiva* studiando il ruolo di modalità di comunicazione diverse: in particolare una modalità di comunicazione semplice e indiretta in cui ciascun robot semplicemente è in grado di percepire la direzione di movimento dei robot posti nelle vicinanze e una modalità di comunicazione più complessa con domanda e risposta, in cui un robot può *richiedere ad altri robot informazioni relative ad uno specifico obiettivo* e *ricevere una risposta congruente alla domanda effettuata*. In questa relazione illustrerò un esperimento in particolare, che utilizza il secondo sistema di comunicazione implementato.

L'esperimento è stato implementato utilizzando il simulatore EvoRobot* accennato nel capitolo precedente di questa relazione, e si è scelto di applicare il processo evolutivo esclusivamente al sistema nervoso dei robot, lasciando invariati il corpo e il sistema senso-motorio.

Ciò che si evolve quindi è la rete neurale, in particolare i pesi delle connessioni sinaptiche e le proprietà dei neuroni interni, lasciando invariata l'architettura dell'intera rete.

È importante puntualizzare che l'esperimento è stato realizzato con "robot-cloni", dotati cioè dello stesso sistema senso-motorio, dello stesso corpo, e dello stesso sistema nervoso, senza la presenza di leader.

In questo esperimento lo swarm è composto da tanti robot E-Puck (accennato nella prima parte di questa relazione) il quale presenta un apparato sensoriale idoneo a soddisfare il fine di questo esperimento. Alcuni dei suoi sensori sono gestiti attraverso le classi del simulatore EvoRobot* (accennate nel capitolo precedente), e per questo esperimento ho implementato delle classi e i metodi idonei alla gestione di *sensori dedicati* i quali caratterizzano il task e i comportamenti mostrati dai robot (tutti i sensori ed attuatori relativi ai robot verranno descritti più avanti nelle sezioni dedicate).

3.2 Caratteristiche e particolarità dell'esperimento

In questo esperimento un insieme di robot viene fatto evolvere per apprendere:

capacità di esplorazione dell'ambiente evitando gli ostacoli e gli scontri tra i

robot, ***capacità di reperimento di alcune risorse situate nell'ambiente*** sfruttando le sue caratteristiche e le sue regolarità, e ***capacità comunicative*** che aiutano i robot a coordinare il movimento dell'intero sciame.

Abbiamo immaginato che per sopravvivere questi robot debbono procurarsi due sostanze: una *sostanza rossa* e una *sostanza blu* le quali in paragone con un ambiente naturale, possono rappresentare due **risorse essenziali** per vivere come per esempio il cibo e l'acqua.

Durante il processo evolutivo i robot vengono valutati sulla base della capacità di reperire periodicamente queste sostanze: *più un robot riesce a soddisfare i propri bisogni, più esso viene premiato e più aumenta la probabilità di trasmettere le proprie informazioni genetiche (e quindi anche fenotipiche) al passo generazionale successivo.*

In particolare quando un robot entra nella *target-area rossa*, istantaneamente porta il suo livello di *sostanza rossa* a 1 (massimo) e appena abbandona quest'area, il livello pian piano decrementa fino ad arrivare a 0. Stesso discorso dicesi per la *sostanza blu* e la *target-area blu.* Il fattore di decremento è uguale per entrambe le sostanze, è scelto a priori dallo sperimentatore, ed è settabile dal file di configurazione dell'esperimento (evorobot.ini), questo perché esso può variare in base alla progettazione dell'ambiente in particolare alla distanza e alla grandezza delle due target area.

É importante considerare che **ogni robot ha bisogno di entrambe le sostanze** e che viene premiato solo quando le ha reperite entrambe. Questo implica che la funzione di fitness di questo esperimento deve essere diversa da 0 esclusivamente quando sia il livello di *sostanza rossa* che il livello di *sostanza blu* sono maggiori di 0. Non è considerato utile ai fini del task soddisfare una sola sostanza ed ignorare l'altra.
Ciò che si aspetta da questa funzione di fitness (rigorosamente legata allo stato metabolico e senza componenti aggiuntive) è che ogni singolo robot mostri dei comportamenti in cui ricerca una sostanza se ne ha bisogno, e una volta trovata e attraversata una delle due target-area, e quindi soddisfatto il bisogno metabolico di una prima sostanza, deve cercare e raggiungere la target-area opposta, prima che il livello della prima sostanza appena assimilata arrivi a 0.

3.3 Il sistema di comunicazione

Il sistema di comunicazione utilizzato in questo esperimento, come accennato sopra, è strutturato e coinvolge uno scambio di informazioni in cui un robot invia una richiesta ad altri robot e in cui questi rispondono al primo in modo congruente. La comunicazione è limitata nello spazio tra robot posti entro una certa distanza che viene settata attraverso un parametro.

Tale sistema di comunicazione è stato realizzato assumendo che il robot che effettua la domanda invia il proprio stato metabolico (livello di sostanza rossa e di sostanza blu al momento della comunicazione) agli altri robot, e questi rispondono con l'azione che farebbero se avessero lo stato metabolico del robot che ha inviato la richiesta.

L'informazione ricevuta, consiste sostanzialmente in *un* ***informazione di navigazione che indica quanto un robot avrebbe modificato la sua rotta verso destra, o verso sinistra, se anziché basarsi sui propri bisogni metabolici, si fosse basato sui bisogni di un altro robot***. Immaginando al posto di due robot, due esseri umani capaci di colloquiare verbalmente (e con buone intenzioni di cooperazione) la comunicazione tenuta in considerazione in questo esperimento si può riassumere (in modo non formale) in uno scambio domanda-risposta tra i due in cui uno dice: *"Io ho queste necessità. Tu cosa faresti al posto mio?"* e l'altro risponde: *"Io al posto tuo proseguirei in questa direzione"*.

La comunicazione tra i robot si può esprimere nei seguenti passi:

- ciascun robot invia agli altri robot posti nelle vicinanze una richiesta contenente le proprie informazioni metaboliche
- i robot che ricevono questo messaggio, sostituiscono questi due valori con i propri valori metabolici e propagano la propria rete neurale
- l'output neurale generato da ogni robot viene convertito in un angolo

(espresso in coordinate assolute dell'ambiente) e inviato al robot che ha iniziato la comunicazione il quale una volta ricevuto tutti i consigli, ne fa una media

- il robot che ha inviato la richiesta riceve il cambiamento di direzione suggerito, in media, dai diversi robot che hanno risposto. Di conseguenza può decidere di modificare il proprio orientamento sia sulla base delle proprie informazioni sensoriali sia sulla base della risposta media ricevuta dagli altri robot.

3.4 Obiettivo

L'obiettivo cercato è quello di dimostrare che un sistema di comunicazione locale di questo tipo, in cui i robot tendono a scambiarsi informazioni utili a soddisfare i loro bisogni, aiuti l'intero sciame a coordinare la navigazione in maniera tale da soddisfare le esigenze metaboliche collettive.

3.5 Caratteristiche di Embodiment

L'esperimento si svolge simulando uno sciame che vede coinvolti una decina robot E-Puck: ecco qui una lista delle caratteristiche del sistema sensoriale e degli attuatori incorporati nei robot utilizzati per questo esperimento

Fig. 14: Robots E-Puck (appena 7 centimetri di diametro, e 5 di altezza per un peso di circa 200g) . Viene utilizzato spesso in esperimenti di **Swarm Robotics**, e **Evolutionary Robotics** grazie al suo costo ridotto e alle sue piccole dimensioni. Per questo esperimento sono stati utilizzati alcuni dei sensori che questo robot offre, e altri sono stati implementati ad hoc.

Sensori di prossimità :

8 sensori ad infrarosso, ognuno dei quali si attiva in maniera inversamente proporzionale alla distanza dell'ostacolo (più l'ostacolo è vicino, più il sensore si attiva). Hanno un raggio di portata di circa 8 centimetri e vengono codificati ognuno con un neurone che si attiva da 0 a 1, in proporzione alla distanza dell'oggetto.

Step Motor :

Entrambi i motori del robot vengono controllati con due neuroni: uno codifica in maniera continua da 0 a 1 la velocità di rotazione delle due ruote: valore 0 = il motore rimane fermo, valore 1 = il motore va alla velocità massima (circa 13 cm/s); un altro neurone codifica in maniera continua la differenza di velocità tra le due ruote: da 0 a 0.5 il robot gira verso sinistra, da 0.5 a 1 il robot gira verso destra, e un valore fisso a 0.5 indica che il robot va dritto. Questa codifica dei motori viene poi utilizzata dal simulatore per assegnare un valore reale di velocità al robot.

Camera RGB :

Videocamera con risoluzione 640x480 che permette al robot di distinguere i colori degli oggetti nell'ambiente. Questo sensore è fondamentale in questo esperimento in quanto permette l'identificazione delle zone target. Il campo visivo offerto da questa camera ha una larghezza focale di 36 gradi diviso in 3 fotorecettori da 12 gradi ciascuno. La risoluzione di questa camera permette di identificare e distinguere solo gli oggetti posti entro una distanza limitata. Viene codificato con un totale di 9 neuroni, 3 per ogni componente di colore (3 neuroni per il Red, 3 per il Green, e 3 per il Blu). Ognuno dei tre neuroni rappresentanti un colore, codifica quale fotorecettore viene attivato, il che dipende dalla posizione dell'oggetto rispetto al robot (un neurone codifica la presenza del colore a destra, uno a sinistra e

uno al centro, rispetto al campo visivo del robot). Ognuno dei nove neuroni si attiva da 0 a 1 in proporzione a quanto è presente il relativo colore nel campo visivo del robot (il che dipende dalla distanza tra robot e oggetto).

Sensore di Ground :

Il ground sensor, posto alla base del robot è un sensore in grado di riconoscere il colore del pavimento. Il colore del pavimento è uniformemente bianco all'interno dell'area quadrata. La colorazione in rosso e in blu delle zone in cui il robot è in grado di assorbire le due sostanze metaboliche è utilizzata solo per semplificare la comprensione del comportamento. Il ground sensor è strettamente ideato per riconoscere le zone dell'ambiente vicino ai muri, (zone che non offrono benefici metabolici) e non si attiva quando il robot passa sopra le zone target (vedere la sezione successiva relativa alle caratteristiche dell'ambiente).

Sensore Metabolico :

Il Sensore Metabolico è stato realizzato ad hoc per questo esperimento e aggiorna il suo stato grazie alle capacità della camera di poter riconosce il colore della risorsa. É codificato con due neuroni i quali si attivano reciprocamente se si raggiunge una delle due zone target. Quando il robot sta in una delle due zone target, il relativo neurone si attiva a 1, e una volta che il robot si allontana dalla zona target, questo

livello di attivazione decrementa di una costante ad ogni step del simulatore. La costante può essere settata a priori nel file di configurazione, in modo da poterla adattare alla dimensione dell'ambiente e alla distanza e alla grandezza delle zone target.

Sensore di Comunicazione :

Il Sensore di comunicazione è stato realizzato ad hoc per questo esperimento. Come descritto brevemente sopra, questo sensore codifica la differenza angolare tra la direzione mantenuta dal robot prima di effettuare la comunicazione, e la direzione acquisita dal resto dello sciame dopo la fase di comunicazione. Più precisamente è codificata tramite due neuroni sensoriali che si attivano da 0 a 1 in maniera mutualmente esclusiva, a seconda se la risposta media fornita degli altri robot suggerisce di girare a destra o a sinistra. Il primo neurone si attiva se la risposta ricevuta suggerisce di girare a sinistra ed ha un livello di attivazione proporzionale alla rotazione necessaria (valore=1 implica una rotazione verso sinistra di 180 gradi). Il secondo neurone si attiva se la risposta ricevuta suggerisce di girare a destra e anch'esso ha un livello di attivazione proporzionale alla rotazione necessaria (valore=1 implica una rotazione verso destra di 180 gradi). Lo stato sensoriale [0, 0] si verifica sia quando il robot non riceve nessuna indicazione sia quando l'indicazione ricevuta consiste nel non variare la propria direzione.

Tutti i neuroni codificanti i sensori e gli attuatori riassunti sopra formano la seguente rete neurale, che rappresenta il sistema nervoso (sistema di controllo o robot-brain) dei robot di questo esperimento :

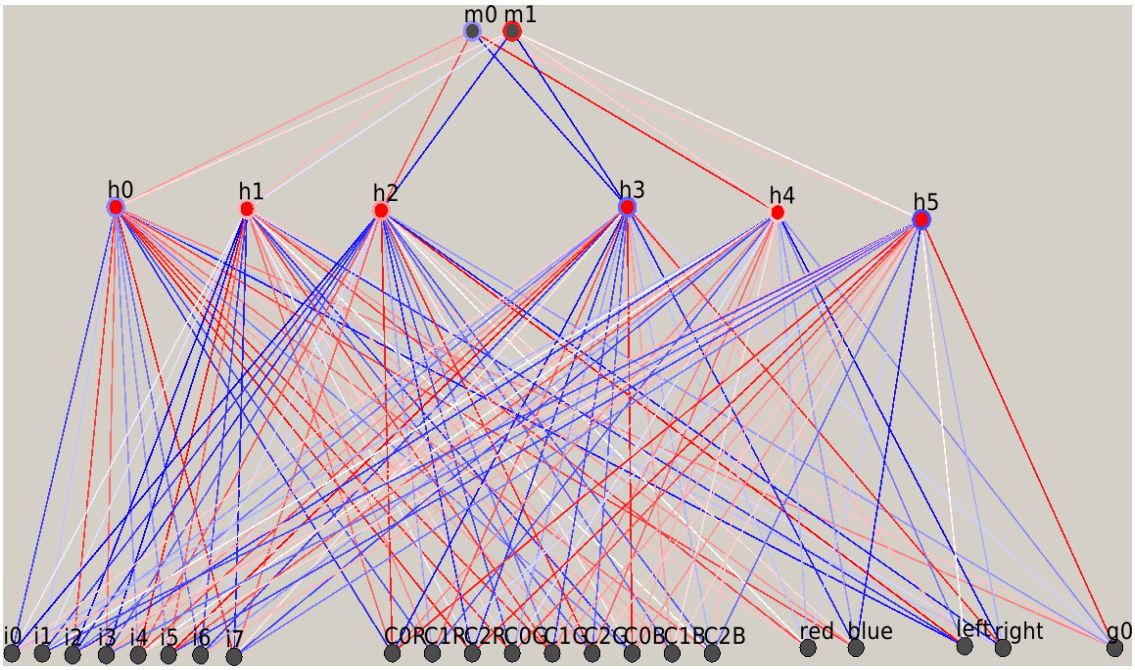

Fig. 15: Il sistema nervoso incorporato nei robot di questo esperimento. Il livello in basso indica lo stato sensoriale di ogni singolo robot, ossia le informazioni acquisibili dall'ambiente, sia fisico che sociale. In particolare il primo gruppo di 8 neuroni codifica le informazioni elaborate dai **sensori di prossimità**, il secondo gruppo di 9 neuroni indica le informazioni ricevute dalla **camera RGB** con 3 neuroni dedicati ad ogni componente di colore, e per ogni componente di colore 3 neuroni che indicano l'orientamento del colore rispetto al campo visivo del robot, il terzo gruppo di 2 neuroni codifica lo **stato metabolico**, ossia lo stato motivazionale dei robot, il quarto gruppo di 2 neuroni indica la codifica delle informazioni ricevute dal **sistema di comunicazione** e l'ultimo neurone indica l'attività del **groundsensor**; data la complessità di questa rete neurale, dovuta all'alto numero di neuroni di input è necessario introdurre uno strato di **neuroni interni** (hidden), che funge da strato intermedio di calcolo.
Questa figura mostra una rete neurale evoluta per questo esperimento: il colore delle sinapsi che legano gli strati rappresenta il grado di eccitamento o di inibizione di ognuna, rispettivamente dal rosso al blu.

3.6 Caratteristiche dell'ambiente e Situatedness

I robot vengono fatti vivere ed evolvere in diverse varianti dell'ambiente mostrato in figura 16.

L'ambiente è un quadrato 3000 x 3000 (millimetri nel mondo reale, PIXEL* SCALA DI RISOLUZIONE nel mondo simulato) formato da due zone target colorate rappresentanti le zone dell'ambiente che contengono le sostanze necessarie ai robot (rispettivamente blu e rossa). I robot vengono lasciati vivere nell'ambiente per 6000 step.

Ad ogni passo generazionale le due zone target assumono posizioni casuali nell'ambiente, e rimangono distanti abbastanza da costringere i robot ad esplorare l'ambiente ma permettergli comunque di passare in entrambe le zone entro il tempo consentito dal sistema metabolico. In questo modo si evita di premiare i robot che assumono strategie troppo "geometriche" ossia che riescono a passare in entrambe le zone target entro il tempo necessario semplicemente "rimbalzando" sui muri, senza avere una visione attiva della camera (comportamento notato dalle prime implementazioni dell'esperimento in cui le due zone rimanevano nella stessa posizione durante tutto il processo evolutivo). Al centro di ogni zona target infatti è posto un oggetto dello stesso colore della risorsa, che rappresenta il vero stimolo sensoriale.

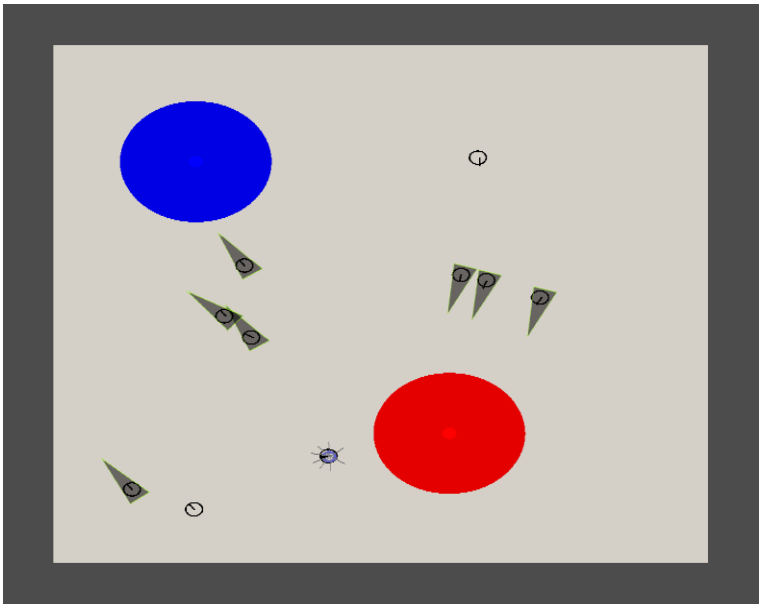

Fig. 16: Ambiente utilizzato in questo esperimento. Presenta due zone target indicanti zone contenenti le risorse metaboliche necessarie ai robot. L'ambiente è un quadrato di lato 3000 e le zone target hanno raggio 300. Ad ogni passo generazionale le due zone target cambiano disposizione in maniera casuale ma garantendo ai robot la possibilità di soddisfare le proprie risorse entro il tempo concesso del sistema metabolico.

Il colore più scuro del pavimento ai bordi dell'ambiente rappresenta un caratteristica aggiunta che permette ai robot di avere una maggiore informazione sensoriale riguardo le zone dell'ambiente che non sono benevoli ai robot. Più in particolare è utile a differenziare l'attività degli infrarossi dovuta dalla presenza di un altro robot nelle vicinanze, da quella dovuta alla presenza di un muro. Anche questa è una scelta sperimentale che aiuta i robot a non apprendere strategie troppo geometriche,

3.7 Dati e statistiche

Al termine del processo evolutivo i robot vengono testati nello stesso ambiente in cui si sono evoluti e inizialmente si registrano i dati della fitness privandoli delle capacità di comunicazione. In questo modo si può vedere meglio l'effetto della comunicazione sia confrontando i dati della fitness, sia osservando il comportamento dei robot nell'ambiente. In entrambi i casi (con / senza le capacità comunicative) il processo evolutivo è stato replicato 4 volte, affidandosi ogni volta ad una sequenza di numeri casuali generata da un seed differente.

I diversi seed portano i robot ad assumere diverse capacità di navigazione a volte molto simili e altre volte completamente differenti. Riporto in seguito il grafico che mostra l'andamento delle 4 repliche dell'esperimento dopo un evoluzione durata 750 generazioni, in particolare l'andamento della fitness assegnata al migliore individuo generazione dopo generazione.

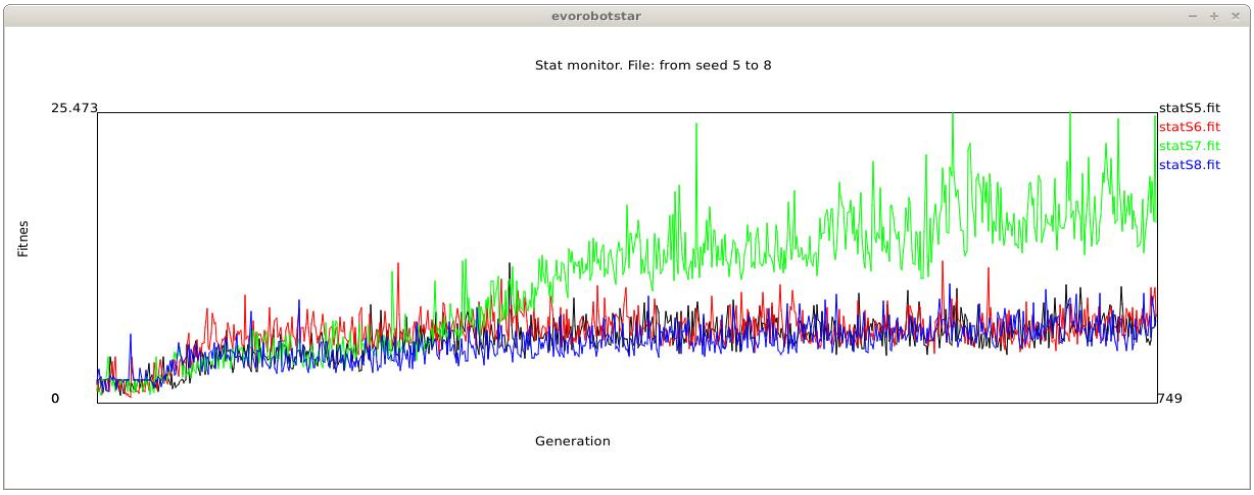

Fig. 17: Grafico della fitness senza le abilità di comunicazione. Il grafico mostra il punteggio (valore di fitness) preso dell'individuo migliore generazione per generazione, delle quattro repliche dell'esperimento. In questo caso il seed 7 (verde) è stato quello che ha portato i risultati migliori, segnando una massima fitness di 25,473 punti.

I migliori risultati acquisiti dai robot senza le abilità comunicative vedono una fitness di **25,473 punti** che rappresenta la somma delle fitness calcolate ad ogni step.

Il prossimo grafico riporta invece i dati della fitness assegnata ai robot con abilità comunicative:

il resto dei parametri tra i due esperimenti è rimasto immutato e anche in questo caso l'esperimento è stato replicato quattro volte con quattro semi di generazione casuale differenti.

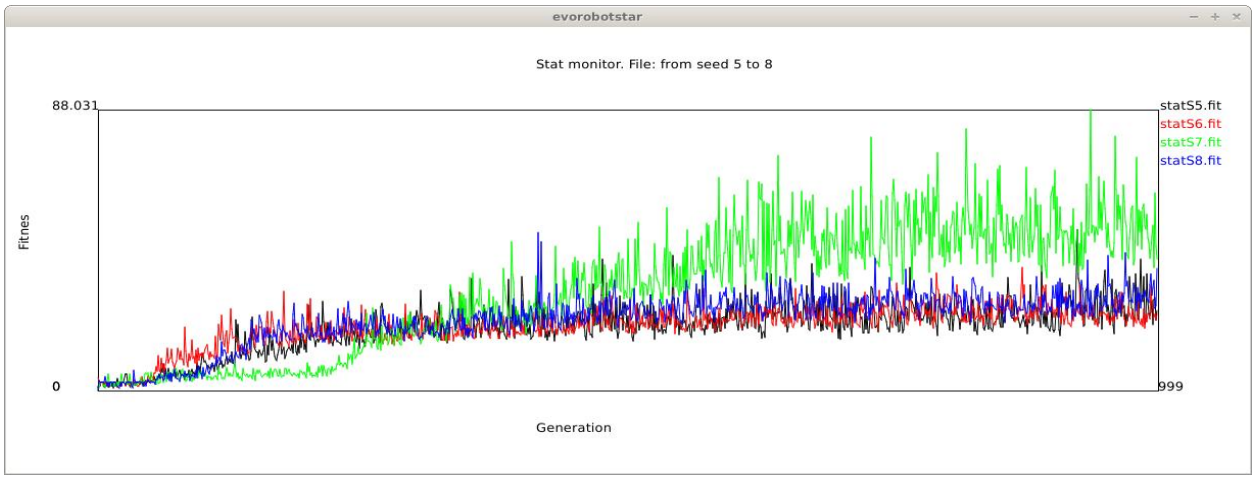

Fig. 18: Grafico della fitness con le abilità di comunicazione. Il grafico mostra il punteggio (valore di fitness) preso dell'individuo migliore generazione per generazione, delle quattro repliche dell'esperimento. Anche in questo caso il seed 7 (verde) è stato quello che ha portato i risultati migliori, segnando una massima fitness di 88,031 punti.

I migliori risultati acquisiti dai robot senza le abilità comunicative vedono una fitness di **88,031 punti** che rappresenta la somma delle fitness calcolate ad ogni step.

La fitness dei robot con le abilità comunicative mostra un incremento superiore al 200% .

3.8 Comportamenti osservati nei robot con capacità comunicative

Il test della capacità dei robot evoluti viene eseguito scegliendo il genotipo del migliore individuo in base al seed che ha portato i migliori risultati. Lasciando vivere i robot nell'ambiente si può osservare una buona adeguatezza a diverse disposizioni spaziali delle zone target nell'ambiente e la strategia vincente vede una "navigazione quasi circolare" con una visione attiva della camera, in cui i robot formano dei piccoli gruppi molto compatti che passano da una zona target ad un altra (a volte i robot si unisco in un unico sciame ma questo non dura per molto tempo). Ogni robot passa sopra entrambe le zone target entro i tempi concessi dal sistema metabolico almeno una volta durante l'esperimento. Con la strategia di navigazione mostrate dai robot, quando lo sciame si muove verso una zona target non tutti i robot riescono sempre a raggiungere l'obiettivo, ma comunque chi non riesce a passare sopra entrambe le zone target, aiuta qualche altro robot a farlo, e molto probabilmente nel prossimo tentativo riuscirà a soddisfare entrambe le sostanze, essendo aiutato da qualche altro robot. In media **ogni robot durante il test vive con uno stato metabolico diverso da 0 per circa il 40% del test.**

Conclusioni ed eventuali sviluppi futuri

Gli esperimenti effettuati durante questo tirocinio hanno dimostrato come la comunicazione possa essere utilizzata per favorire il successo di un task, in particolare questo tipo di comunicazione che comprende lo scambio di informazioni motivazionali riguardanti i bisogni metabolici dei robot e la risposta relativa, hanno dato ai robot la possibilità di coordinarsi ed effettuare una navigazione collettiva soddisfando i bisogni metabolici di ognuno per una buona parte del test.

Durante questo tirocinio è stata esaminata la possibilità di implementare altre diverse varianti di questo esperimento: una di queste comprende una comunicazione del tipo "tutti con tutti" senza limiti di distanza (tutti chiedono e tutti rispondono), un altra variante sempre del tipo "tutti con tutti" in cui i robot chiedono consiglio a tutti ma solo chi riesce a vedere la zona contenente le sostanza metaboliche risponde alla comunicazione, ed un'altra variante più interessante che vede l'implementazione di un ulteriore neurone che fa scegliere al robot se partecipare alla comunicazione oppure no.

Ulteriori sviluppi dal punto di vista software possono riguardare l'implementazione di ulteriori livelli di parallelizzazione, lasciando scegliere allo sperimentatore su quali parametri basare parallelizzazione per il proprio esperimento, non solo per quanto riguarda l'algoritmo genetico ma anche per

quanto concerne la fase di test, che in un contesto multi-robot può essere molto utile.

Ringraziamenti

Ringrazio tutte le persone che hanno reso possibile e piacevole questo mio percorso universitario, in primis la mia famiglia per l'immancabile appoggio da qualsiasi punto di vista, gli amici vicini e gli amici lontani.

Ringrazio il prof. Andrea Sterbini e il prof. Stefano Nolfi per l'occasione concessami, per avermi fatto gustare il sapore della ricerca e per la cordiale assistenza ricevuta.

Ringrazio il team del LARAL-ISTC-CNR per aver reso questa esperienza di tirocinio oltre che interessante, piacevole e accogliente.

Bibliografia

[1] Stefano Nolfi (ISTC-CNR) "Introduzione alla robotica autonoma: Comportamento e Cognizione come Sistemi Adattivi Complessi"

[2] Erol Sahin (KOVAN Dept. of Computer Eng.) "Swarm Robotics: From Sources of Inspiration to Domains of Application"

[3] Marco Mirolli, Stefano Nolfi (ISTC-CNR) "Evolving communication in embodied agents: Theory, Methods and Evaluation "

[4] Vito Trianni, Stefano Nolfi (IRIDIA CoDE , ISTC-CNR) "Engineering the Evolution of Self-Organising Behaviours in Swarm Robotics: A Case Study"

[5] Dario Floreano "Manuale sulle reti neurali"