POLITECNICO DI MILANO
Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

STRATEGIE EVOLUTIVE PER COMPITI DI APPRENDIMENTO SOCIALE IN ROBOT AUTONOMI

Relatore: Prof. Andrea Bonarini
Correlatore: Dott. Elio Tuci

Autore:
Pini Giovanni, matricola 668032

Anno Accademico 2006-2007

Alla mia famiglia.
Ai miei amici.

Sommario

Il lavoro presentato in questa tesi riguarda una branca relativamente giovane della robotica: la robotica evolutiva (*Evolutionary Robotics - ER*), in questo caso applicata a robot mobili. La tematica specifica qui affrontata é lo studio di fenomeni di apprendimento, ovvero quei casi in cui si osserva un cambiamento del comportamento di un individuo nel medio-lungo periodo a causa di esperienze.
Il lavoro proposto riguarda due tipologie diverse di apprendimento: individuale e sociale. La prima é dovuta alle esperienze individuali dell'agente, che impara attraverso l'interazione con l'ambiente. Nel secondo caso invece il cambiamento nel modo di agire non é dettato unicamente dall'interazione con il mondo, ma deriva sopratutto dall'osservazione del comportamento di altri agenti.
Scopo della tesi é mostrare che attraverso l'evoluzione artificiale é possibile ottenere un controllore neurale che consenta ad un agente autonomo di poter apprendere nelle due diverse modalitá.
Per ottenere questi risultati é stato implementato un simulatore per i robot *e-puck*, che ha costituito lo strumento principale per l'esecuzione degli esperimenti ed ha permesso di esplorare strategie per ottenere il comportamento voluto. In parallelo allo sviluppo del software sono stati condotti test sui robot per validare i modelli costruiti ed i controllori evoluti. Una volta ottenuta una piattaforma affidabile si é passati alla fase di ricerca vera e propria, cominciando dalla specifica di un compito che consentisse di osservare i fenomeni di apprendimento. Infine sono state condotte delle simulazioni per evolvere la rete neurale, investigando diverse tecniche per ottenere il comportamento desiderato.
Il lavoro ha portato all'evoluzione di un controllore neurale in grado di affrontare compiti che richiedono capacitá di apprendimento individuale e sociale. In letteratura non si trovano esempi di lavori in cui l'approccio evolutivo viene utilizzato per ottenere controllori neurali che siano in grado di

apprendere socialmente, scopo della tesi è mostrarne l'applicabilità in questo ambito.

Ringraziamenti

Ringrazio i miei genitori, mia sorella Anna e i miei parenti, che mi hanno sempre aiutato e sostenuto.
Ai miei amici: Alessandra, Andrea, Beluf, Berte, Ceco, Chiara, Cippi, Ciri, Dagno, Dario, Della, Denu, Diego, Ellen, Fiore, Gabriele, Gabro, Lisa, Luca, Mauri, Nunzio, Pugno, Rudy, Saldo, Scott, Teo. Grazie per i bei momenti passati assieme.
Grazie ed in bocca al lupo ad Ale e Miriam, la mia famiglia milanese.
Grazie a Elisa, Fede, Mario e Stefano.
Un grazie a Elio Tuci per la supervisione del mio lavoro e per l'aiuto datomi e a Marco Dorigo per avermi dato la possibilità di lavorare a IRIDIA.
Grazie al professor Andrea Bonarini per la sua disponibilità.
Grazie a Clara dello studesk2 per la sua gentilezza.
To e-puck group guys: Alex, Antoine, Francesco, Laurent, Mouhcine, Mauro, Olivier, Shervin, we had a lot of fun together (even with Shervin).
Thanks to all IRIDIA people: Anders, Carlo, Carlotta, Christos, Francisco, Marco, Muriel, Max, Prasanna, Rehan, Thomas, Xavier.
To all Erasmus friends: Alvaro, Andres, Andrew, Arminda, Aymeric, Bertha, Bozan, Carlos, Cesar, Chiara, Dave, Dominik, Emilia, Gracia, Helena, Irene, Jack, Jo, Joanna, Joni, Julia, Kashia, Kate, Laura, Liz, Lola, Lucas, Luisa, Merijn, Michela, Mounir, Nick, Oscar, Paola, Rachel, Rowen, Susan. I will never forget you, hope to see again all of you.

Indice

G. Pini, E. Tuci, and M. Dorigo, *Evolution of social and individual learning in autonomous robots*, European Conference on Artificial Life, 2007.

Capitolo 1

Introduzione

Obiettivo di questo lavoro è l'implementazione di controllori per robot autonomi, utilizzando la robotica evolutiva come strumento di sviluppo. In particolare in questa tesi viene preso in esame il tema dell'apprendimento: ai robot è richiesta la capacità di adattare il proprio comportamento sulla base di esperienze. Nel caso studiato i robot sono coinvolti in un semplice compito: muoversi rispetto ad una sorgente luminosa e adattare il tipo di movimento sulla base di determinati eventi. L'apprendimento si manifesta nel fatto che i robot siano in grado di associare il tipo di movimento al tipo di evento percepito.
Vengono prese in esame due tipologie di apprendimento: individuale e sociale in cui, rispettivamente, il robot apprende in modo autonomo attraverso l'interazione con l'ambiente oppure attraverso l'interazione con un altro robot che ha già appreso precedentemente.
Il lavoro ha portato allo sviluppo di un simulatore per i robot e-puck, attraverso il quale è stato possibile ottenere un controllore neurale ricorrente, creato attraverso tecniche di evoluzione artificiale, in grado di manifestare entrambe le tipologie di apprendimento. I risultati ottenuti sono contenuti anche in un articolo, allegato alla tesi, presentato alla conferenza ECAL 2007 (European Conference on Artificial Life - Lisbona, Portogallo).
Il capitolo seguente riassume lo stato dell'arte per quanto riguarda la robotica evolutiva, l'uso delle reti neurali in questo contesto e gli aspetti legati all'apprendimento. Il secondo capitolo contiene una descrizione degli strumenti più utilizzati in robotica evolutiva: reti neurali e algoritmi genetici. Il terzo capitolo presenta i componenti utilizzati per condurre la ricerca: il robot e-puck e un simulatore. Il quarto capitolo descrive l'esperimento preso in esame e le strategie evolutive utilizzate per ottenere i controllori,

indicando di volta in volta i risultati ottenuti. Infine il quinto capitolo trae le conclusioni, attraverso un analisi qualitativa dei risultati e proponendo possibili estensioni al lavoro.

Capitolo 2

Stato dell'arte

Questo capitolo descrive lo stato dell'arte per quanto riguarda la robotica evolutiva e gli studi legati all'apprendimento in ambito robotico. Il termine robotica evolutiva indica un approccio automatizzato all'implementazione di controllori robotici, ispirato all'evoluzione naturale. Approcci più "tradizionali" presentano una serie di aspetti che rendono ostica la progettazione e la costruzione di sistemi di controllo per agenti autonomi.

2.1 I perché dell'approccio evolutivo

Come fanno notare Nolfi e Floreano [26], oggi sono presenti numerosi agenti in grado di affrontare compiti molto complessi quali battere il campione del mondo di scacchi o risolvere problemi di ottimizzazione, ma non ci sono robot mobili intelligenti nelle nostre case ed uffici. La complessità nel design di robot mobili deriva dal fatto che il comportamento di tali agenti emerge dall'interazione col mondo esterno. Agente e mondo possono essere visti come un solo sistema in quanto ogni azione motoria ha due effetti: definisce quanto bene il robot si stia comportando in relazione all'obiettivo da raggiungere ed inoltre influenza lo stato in cui si trova il mondo. A stati diversi corrispondono percezioni sensoriali diverse le quali, sulla base dell'obiettivo da raggiungere, portano a differenti scelte da parte del robot. Il comportamento che si osserva è quindi profondamente correlato al mondo nel quale il robot agisce.
Ciò consente di affrontare compiti complessi con agenti semplici che sfruttano caratteristiche ambientali, ma rende difficile la definizione di regole che consentano di ottenere determinati obiettivi. Questo perché occorre tenere

conto sia dell'obiettivo che si vuole perseguire, ma anche del fatto che le azioni hanno conseguenze nel lungo termine che potrebbero pregiudicare la riuscita del compito e l'applicabilità di altre azioni. L'incertezza nei confronti dell'ambiente, sia in termini di caratteristiche non note in maniera precisa, sia in termini di effetti delle azioni, rappresenta, assieme alla variabilità delle caratteristiche ambientali, il problema più ostico nell'ingegnerizzazione di sistemi di controllo per agenti autonomi.
L'approccio evolutivo ridimensiona questo aspetto in quanto lo sviluppo dei controllori si basa su un processo di valutazione del comportamento globale del sistema nel suo ambiente. Con accorgimenti quali l'introduzione di rumore, nel caso ci si avvalga di simulatori, o direttamente attraverso la valutazione nel mondo reale si possono ottenere sistemi di controllo robusti e con buone capacità di generalizzazione.
Pratica tipica in campo ingegneristico è l'affrontare problemi non banali col paradigma *divide et impera*: la soluzione di un problema deriva da una decomposizione in sottoproblemi che vengono studiati separatamente per poi combinarne le soluzioni. Anche se questa metodologia può portare a risultati soddisfacenti presenta alcuni aspetti critici difficilmente aggirabili [21]. Anzitutto non è sempre chiaro come suddividere un problema in parti più semplici, individuando elementi indipendenti da poter trattare con soluzioni specifiche. Il numero di potenziali interdipendenze tra i componenti cresce in modo esponenziale col numero di moduli; pertanto esiste un limite alla quantità di sottosistemi implementabili, oltre il quale la complessità dovuta alla gestione delle interazioni risulta paragonabile a quella della risoluzione del problema originario e quindi l'approccio *divide et impera* perde di efficacia. Il problema della gestione delle interdipendenze è amplificato dal fatto che alcune non sono dirette, ma mediate dal mondo esterno e pertanto più difficili da individuare ed affrontare.
L'approccio classico era di dividere il sistema in tre parti logiche che effettuassero le funzioni di pianificazione, percezione ed azione; questa metodologia ha subito critiche da numerosi ricercatori circa la sua validità.
Un altro modo di affrontare il problema è stato proposto da Brooks [7]. L'idea è suddividere il sistema sulla base del comportamento che si vuole ottenere: si implementano una serie di moduli che realizzano comportamenti di base e un modulo che li coordina per definire il comportamento a livello globale. Usando questo approccio resta comunque al progettista il compito di suddivisione del sistema. Il vantaggio è nel fatto che l'agente non deve costruire un piano d'azione per perseguire i propri obiettivi, quindi si ridimensianano gli aspetti problematici legati all'incertezza sugli effetti delle azioni, dai quali dipende la possibilità di attuare il piano costruito.

Ha senso parlare di un determinato comportamento solo nel momento in cui questo si manifesta attraverso l'interazione con un ambiente. Dato un controllore è difficile predire il comportamento che si potrà osservare una volta che questo guida un agente. È però vero anche il contrario: dato un comportamento desiderato è difficile dire quale controllore sia il migliore per ottenerlo, ovvero progettare un sistema di controllo che lo implementi.
In campo di robotica evolutiva sono state adottate differenti metodologie, che si differenziano l'una dall'altra in termini di strutture manipolate dall'evoluzione artificiale, che codificano il problema affrontato, e algoritmo di manipolazione di tali strutture, che costruisce la soluzione al problema. Alcuni autori hanno proposto lavori in cui i controllori evolvono direttamente in forma di programmi scritti in qualche linguaggio di alto livello [8]. In questi casi è lasciata al progettista la scelta del modo di rappresentare il programma con strutture dati manipolabili e delle primitive di base utilizzare; questi due aspetti caratterizzano dimensione e struttura dello spazio di ricerca e quindi influenzano la bontà delle soluzioni individuate. Con questo tipo di approccio rimane comunque consistente la parte di lavoro non automatizzata e quindi i potenziali problemi dovuti a cattive scelte progettuali.
In altri casi, il compito dell'evoluzione è quello di trovare un buon insieme di parametri per impostare controllori la cui struttura è predefinta: ad esempio in un lavoro di Dorigo e Schnepf [14], un robot simulato veniva controllato da *learning classifier system* organizzati in modo gerachico; il sistema di classificazione evolveva sostituendo via via alcune regole con delle nuove. Anche qui resta di importanza critica la qualità del lavoro di progettazione.
Esistono persino esempi in cui il controllore evolve direttamente a livello di hardware: usando una FPGA come controllore Thompson [32] riuscì ad evolvere un microcircuito che guidasse il robot evitando di collidere con i muri.
L'approccio più seguito in ambito di evolutionary robotics è però quello di utilizzare reti neurali artificiali (*Artificial Neural Networks, ANN*) come controllori, individuando i parametri per impostare la rete attraverso l'algoritmo evolutivo. Questo modo di affrontare il problema riduce al minimo le difficoltà legate agli aspetti critici descritti in precedenza. Anzitutto non c'è bisogno, almeno nei casi più semplici, che il controllore sia scomposto in moduli, quindi vengono eliminati i rischi di fallimento e le inefficienze dovute a cattive suddivisioni. Se anche fosse necessaria una scomposizione modulare, l'uso di tecniche evolutive semplifica il problema di gestione delle interazioni tra moduli, che come detto può essere molto ostico. L'unica scelta che resta libera, nei casi più generali, è nella tipologia e architettura della rete; tuttavia esistono lavori in cui anche l'individuazione della

topologia del controllore o addirittura della morfologia dei robot è lasciata all'evoluzione. La motivazione di questa scelta deriva dalla comparazione con l'evoluzione naturale: esattamente come certi comportamenti risultano favorire una specie e tendono quindi a essere mantenuti durante l'evoluzione, anche quegli apparati senso-motori che consentono una miglior percezione e azione rispetto al mondo costituiscono un vantaggio per la specie e pertanto tenderanno a conservarsi nelle generazioni [9]. Il fatto che nella maggior parte dei lavori solo lo sviluppo del controllore, e non l'intera morfologia del robot, sia lasciato all'evoluzione deriva dalla necessità di utilizzare determinati modelli di robot, dotati di sensori ed attuatori che non possono ovviamente essere modificati.
Al di là di una semplificazione per chi deve implementare il sistema di controllo, le reti artificiali hanno una serie di pregi che le rendono preferibili rispetto ad altre architetture:

1. Sono robuste nei confronti del rumore. Variazioni lievi dei segnali di ingresso non comportano, in genere, un cambiamento drastico del comportamento della rete. Questa proprietà è ideale per i robot mobili, in quanto utilizzano sensori rumorosi in ambienti rumorosi;

2. Piccole variazioni dei parametri della rete comportano, nella maggior parte dei casi, cambiamenti lievi nel comportamento del robot. Nel caso in cui l'algoritmo utilizzato per condurre l'evoluzione sia un algoritmo genetico, questa proprietà semplifica il compito di ricerca nello spazio delle soluzioni;

3. Possono gestire input di tipo continuo e produrre direttamente un output continuo o discreto;

4. Lavorano con informazioni di basso livello, ideale per molti casi di robotica evolutiva in cui si sfruttano agenti semplici per ottenere comportamenti complessi sfruttando gruppi di robot o il loro interagire con l'ambiente;

Queste proprietà giustificano il fatto che le reti neurali siano diventate uno dei modelli più diffusi in ambito di robotica evolutiva.
Negli ultimi anni è emersa la valenza teorica dell'approccio evolutivo alla robotica: la tecnica non è usata tanto per produrre artefatti utili dal punto di vista tecnologico o pratico, ma per condurre esperimenti scientifici con lo scopo di studiare e fornire "proof of concept" su fenomeni legati al comportamento umano e animale. Il ragionamento alla base di questo approccio

nell'uso delle tecniche evolutive è che la storia della specie umana è relativamente breve rispetto a quella di altre forme di vita [20]; pertanto, in accordo con la teoria evolutiva, le capacità che mostra l'uomo sono costruite su quelle di antenati più semplici. Da questo punto di vista ha senso studiare come certi fenomeni possano emergere in una popolazione di individui semplici e come ciò si leghi all'evoluzione. La robotica evolutiva, combinata con l'utilizzo di reti neurali, si candida come miglior strumento per questo tipo di studi. Il motivo principale è che l'uso delle reti neurali consente di lavorare direttamente con informazioni grezze, lasciando all'evoluzione il compito di sviluppare meccanismi comportamentali di alto livello. Ciò consente di individuare legami tra il processo evolutivo e il risultato finale dell'evoluzione. Studiando quali elementi del modello favoriscono l'emergere di meccanismi senso-motori e cognitivi alla base di certi comportamenti è possibile valutare la validità di ipotesi relative a comportamenti umani e animali [35]. Spesse volte infatti i dati raccolti su esperimenti riguardanti esseri viventi aprono la strada a diverse teorie relative ai meccanismi alla base di un comportamento; esperimenti di robotica evolutiva svolti in condizioni analoghe possono portare al rifiuto di alcune delle ipotesi e suggerirne di nuove [17]. Altra caratteristica dell'approccio evolutivo è l'enfasi sul fatto di minimizzare l'intervento del progettista nel processo, proprietà fondamentale quando si vogliano studiare fenomeni da un punto di vista scientifico. Infine l'uso di ambienti ed agenti artificiali o virtuali comporta maggior velocità nella conduzione di esperimenti e maggior controllo delle caratteristiche ambientali.

2.2 L'apprendimento

Sempre più ricercatori si sono concentrati negli ultimi tempi sul tema dell'apprendimento, ovvero quei casi in cui un individuo modifica autonomamente il proprio comportamento in funzione di una esperienza e mantiene tale variazione nel medio-lungo periodo. I motivi di questo interesse sono diversi e coinvolgono più discipline. I biologi studiano il legame tra apprendimento ed evoluzione, cercando di capire come il primo influenzi la seconda. Entrambi modificano il modo di comportarsi di un individuo, ma intervengono su scale temporali e su elementi diversi: l'apprendimento coglie cambiamenti nell'arco della vita di un individuo e agisce a livello di fenotipo, l'evoluzione adatta la specie a quelle modifiche che occorrono in tempi più lunghi e agisce sul genotipo [25, 16]. Baldwin [1], spiega come la capacità di apprendere un certo modo di comportarsi porti l'evoluzione a selezionare individui in cui

tale comportamento risulta innato (effetto Baldwin, vedere anche [16, 25]). Dal punto di vista ingegneristico interessa il fatto che il robot possa, in maniera autonoma e permanente, cambiare il modo di porsi nei confronti dell'ambiente e pertanto far fronte a situazioni diverse con strategie diverse. Un sistema con questa capacità presenta maggior flessibilità, robustezza ed affidabilità in quanto ha opzioni differenti per raggiungere gli obiettivi per cui è costruito.
Una forma di apprendimento è l'apprendimento sociale (*social learning*) ed è riferito a quei casi in cui un individuo modifica il modo di porsi nei confronti dell'ambiente a seguito dell'interazione con i propri simili; apprendere socialmente significa estrarre conoscenza dall'interagire con gli altri. Questo tipo di comportamento si osserva in natura in molte specie animali: i ratti della famiglia *Rattus Norvegicus* annusano conspecifici per stabilire quali cibi abbiano ingerito e mostrano preferenza per tali alimenti [18, 24]; il guppy (*Poecilia Reticulata*, noto anche come pesce milione) apprende da conspecifici informazioni circa fonti di cibo e potenziali predatori [31]; il macaco Rhesus (*Macaca Mulatta*) può apprendere a fuggire determinati predatori semplicemente osservando video che mostrano conspecifici fare lo stesso [11]. Comportamenti analoghi si hanno nei merli *Merula Turdus* e in alcune specie di pesci. Infine l'uomo, l'esempio più evidente di animale sociale, in cui la maggior parte della conoscenza deriva da imitazione (sopratutto nei bambini) o trasmissione tramite il linguaggio.
Alcuni studiosi descrivono diverse forme di apprendimento sociale, che vanno da casi semplici in cui l'interazione con un conspecifico sposta l'attenzione su elementi del mondo cruciali, aumentando pertanto la probabilità di apprendere, fino al caso più complesso che è l'imitazione. Quest'ultima è definita come "apprendere a compiere un'azione vedendola compiuta da altri" e rispetto ad altre forme di apprendimento sociale richiede capacità cognitive complesse.
Dal punto di vista biologico l'apprendimento sociale rappresenta un vantaggio selettivo: un essere vivente può agire sulla base di comportamenti innati, ma in caso di cambiamenti repentini dell'ambiente questo può risultare poco flessibile; un'alternativa è modificare il comportamento attraverso l'apprendimento individuale, ma ciò comporta la possibilità di compiere scelte errate e pagarne le conseguenze. L'apprendimento sociale è un buon compromesso in quanto il comportamento dei conspecifici è influenzato dalle conseguenze delle loro azioni (un individuo si comporta in un certo modo perché ha una risposta positiva); pertanto apprendere dagli altri porta, nella maggioranza dei casi, a buoni risultati [36]. Gli psicologi sono interessati ai meccanismi alla base dell'imitazione, in particolare interessa sapere se e

come l'osservatore sia in grado di capire le intenzioni dell'individuo osservato e quindi se l'imitare sia un atto intenzionale oppure istintivo; specialmente in quei casi in cui le percezioni derivanti dal compiere un'azione sono diverse da quelle che si hanno quando la si osserva in un altro. Dal punto di vista ingegneristico un sistema in grado di apprendere tramite l'interazione con un "esperto" presenta la flessibilità dovuta all'apprendimento e ha maggior efficenza: la conoscenza si diffonde nel gruppo in maniera più rapida, inoltre si riducono le probabilità d'errore rispetto all'apprendimento individuale.

2.3 Apprendimento in robotica

In materia di robotica ed agenti autonomi molti lavori hanno avuto come tema centrale l'apprendimento sociale. Alcuni lavori si sono concentrati sui casi in cui il soggetto imitato è un essere umano [23, 12], in modo da semplificare la programmazione dei robot usando la dimostrazione o per avere robot che interagiscano con gli esseri umani [6]. In altri casi il sistema esperto è un altro robot [12, 19] o un agente software più o meno sofisticato [24, 2, 5].
Nel caso della robotica evolutiva il problema di sviluppare robot in grado di apprendere (individualmente) viene affrontato tipicamente utilizzando reti neurali di tipo plastico (*Plastic Neural Networks, PNN*), i cui pesi sinaptici cambiano secondo regole simili all'apprendimento hebbiano. Alcuni esempi di uso delle PNN per compiti di apprendimento sono [3, 28, 16, 13]. In altri casi [33, 4, 29] vengono invece usate reti di tipo ricorrente (*Recurrent Neural Networks, RNN*). Per una descrizione dettagliata dei due tipi di rete vedere la sezione 3.1. Secondo alcuni studiosi si può parlare di apprendimento solo nel caso di *PNN*, mentre nelle *RNN* si parla di "memoria". La definizione usata dipende molto dal campo di studi: da un punto di vista biologico e computazionale l'apprendere è associato ad un cambiamento nei pesi sinaptici. Il lavoro qui presentato usa una definizione meno "rigida" di apprendimento, ispirata all'approccio comportamentista alla psicologia: apprendere altro non è che un particolare tipo di comportamento, pertanto esso si manifesta nell'interazione tra un individuo e un ambiente, con dinamiche che si sviluppano su scale temporali medio-lunghe [29]. Apprendere viene inteso come aquisire informazioni sull'ambiente, modificando il comportamento sulla base di questa conoscenza e mantenendo questo cambiamento nel modo di agire anche quando le informazioni che l'hanno provocato vengono a mancare. Da questo punto di vista sia le *PNN* sia le *RNN* possono mostrare capacità di apprendimento; entrambe sfruttano un cambiamento di stato nel

tempo per modificare il comportamento. La differenza è che mentre nelle reti plastiche il concetto di stato è incorporato nei pesi sinaptici, nelle reti non plastiche è nel livello di attivazione dei neuroni. In un esperimento, Blynel e Floreano [3], testano le capacità di apprendimento e di generalizzazione dei due tipi di rete. Nel loro lavoro un robot doveva apprendere la relazione tra un'area obiettivo e dei landmark presenti nel mondo. Analizzando i risultati gli autori giungono alla conclusione che entrambe le tipologie di rete sono in grado di svolgere il compito e quindi di apprendere, ma le reti plastiche mostrano miglior capacità di generalizzazione e possono far fronte a situazioni non incontrate durante l'evoluzione. Tuci e Quinn riprendono questo lavoro in modo critico in [34], ipotizzando che le conclusioni tratte siano dovute alla semplicità del compito studiato; nel loro lavoro i risultati cui giungono hanno mostrato che, nel compito preso in esame, le capacità di apprendimento delle *RNN* sono molto superiori rispetto alle *PNN*. Gli stessi autori in [33] evolvono con successo una rete di tipo ricorrente che mostra capacità di apprendimento. Phattanasri, Chiel e Beer in [29] usano reti ricorrenti per agenti simulati che devono appredere la relazione tra l'odore di un cibo ed il fatto che esso sia commestibile o velenoso.
Dal momento che in letteratura non esistono esempi di applicazione della robotica evolutiva in ambito di apprendimento sociale si è deciso, visti i risultati positivi ottenuti nei lavori citati, di utilizzare controllori ricorrenti. Questa scelta è motivata anche dal fatto che, mentre nelle *PNN* esistono regole specifiche che definiscono come la rete apprende, nelle *RNN* non c'è un meccanismo dedicato, che implementi le modalità di apprendimento, ma viene lasciato all'evoluzione il compito di costruire tali capacità. Ciò è in sintonia con l'approccio evolutivo, che vuole ridurre al minimo l'intervento del progettista nella costruzione della soluzione.
Il lavoro di tesi presentato ha come scopo l'evoluzione di una rete ricorrente per controllare robot autonomi in un compito che richiede capacità di apprendimento individuale e sociale. Il caso studiato è ispirato ad un lavoro di Di Paolo [13] e prevede che il robot apprenda come muoversi nei confronti di una sorgente luminosa. L'apprendimento individuale coinvolge un solo agente che impara il modo di comportarsi attraverso la percezione di un suono; in ambito sociale questo avviene tramite "osservazione" di un esperto che mostra come muoversi rispetto alla luce. Di Paolo usa sia reti ricorrenti che plastiche, ma il suo lavoro riguarda solo l'apprendimento individuale.

Capitolo 3

Gli strumenti della robotica evolutiva

Questo capitolo mira a spiegare, senza la pretesa di essere esaustivo, alcuni degli strumenti utilizzati nella robotica evolutiva.
Come accennato nel capitolo introduttivo, la maggior parte dei lavori in questo campo coinvolgono l'uso di reti neurali in quanto controllori di basso livello, che consentono all'algoritmo genetico di individuare soluzioni libere da preconcetti dello sperimentatore. Le sezioni seguenti descrivono reti neurali e algoritmi genetici.

3.1 Le reti neurali

Le reti neurali sono un modello computazionale che prende spunto dalla biologia. Nella maggior parte degli esseri viventi esistono complesse configurazioni di cellule nervose che hanno il compito di riconoscere stimoli ambientali, elaborati dai sensi, ed attuare delle reazioni in risposta. Le reazioni a stimoli esterni, se viste a livello macroscopico, definiscono il comportamento dell'individuo. Ogni rete è composta da un numero più o meno elevato di elementi di base chiamati neuroni; questi sono connessi gli uni agli altri tramite le sinapsi, che consentono la trasmissione degli impulsi nervosi tra neuroni. Il neurone integra i segnali in ingresso, se questo valore supera una certa soglia allora il neurone produce un segnale di uscita, che a sua volta può fare da ingresso ad altri neuroni. Le reti artificiali modellizzano in modo semplificato i neuroni e le loro interconnessioni; esistono numerosi tipi di modelli, in tutti i casi un neurone implementa una funzione del tipo:

$$y_j = \sigma\left(\sum_{i=0}^{N} \omega_{ji} \cdot x_i\right), \tag{3.1}$$

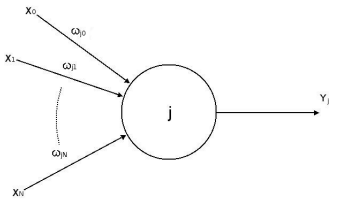


Figura 3.1: neurone artificiale

dove, con riferimento alla figura 3.1, y_j è l'uscita del neurone j, σ la sua funzione di trasferimento, ω_{ji} il peso sinaptico che collega il neurone j ad i ed x_i l'ingresso i-esimo (che può arrivare da un altro neurone o essere esterno). In robotica evolutiva, i neuroni formano delle reti che costituiscono il controllore del robot, lo strato di neuroni in ingresso riceve i dati direttamente dai sensori, tipicamente normalizzati tra 0 ed 1, mentre lo strato d'uscita è connesso con gli attuatori. A seconda del tipo di neurone utilizzato e di come i neuroni vengono connessi tra loro si ottengono tipi diversi di rete; la scelta del modello di neurone da utilizzare dipende dal compito affrontato. Per quanto riguarda la morfologia di rete invece, non ci sono regole che permettano di preferirne una rispetto ad un'altra.
Le reti più semplici sono quelle di tipo *feed-forward* (vedere fig 3.2), in cui i segnali uscenti dai neuroni possono essere trasmessi solo verso strati più a valle e non si hanno quindi delle retroazioni.

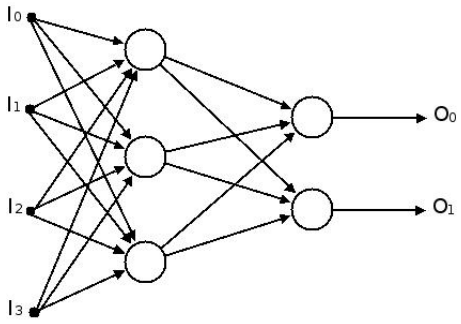


Figura 3.2: Rete feedforward con 4 ingressi I_i $i \in$ [0,3] e 2 uscite O_i $i \in$ [0,1]

In queste reti l'uscita di ogni neurone viene calcolata come:

$$y_j = \sigma\left(\sum_{i=0}^{N} \omega_{ji} \cdot x_i\right) \text{ tipicamente } \sigma(x) = \frac{1}{1+e^{-x}}\ , \tag{3.2}$$

Il significato dei simboli usati è lo stesso dell'equazione 3.1.
Reti di questo tipo sono utilizzate per i compiti che non richiedano memoria di tipo interno, quali ad esempio evitare ostacoli e seguire muri o gradienti luminosi. Ovvero tutti quei compiti puramente reattivi, in cui l'uscita della rete dipende unicamente dai valori che le si presentano in ingresso. Infatti, i neuroni per reti feedforward non incorporano nessun meccanismo che implementi un concetto di stato, necessario affinchè la rete sia in grado di memorizzare informazioni. Compito dell'algoritmo evolutivo è in questo caso quello di individuare un insieme di pesi ω_{ij} per implementare la relazione ingresso-uscita desiderata.
Reti più complesse sono quelle ricorrenti (*Continuous Time Recurrent Neural Netwroks, CTRNN*), in cui l'uscita in un certo istante non dipende solo dagli ingressi correnti, ma anche da quelli passati. Questa proprietà si ottiene collegando i neuroni (che inglobino un meccanismo di stato) non solo dallo strato di ingresso verso quello d'uscita, ma anche in modo apposto, retroazionando alcune delle uscite (vedere fig. 3.3).

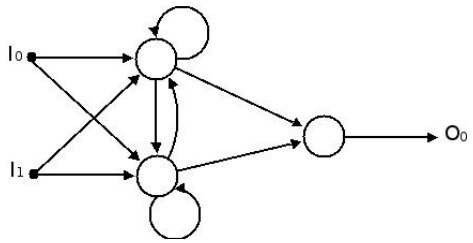


Figura 3.3: Rete ricorrente con 2 ingressi I_i $i \in$ [0,1] e 1 uscita O_0. La rete è ricorrente in quanto i neuroni dello strato di input non sono collegati solo con quelli degli strati successivi, ma anche tra loro.

Le equazioni che regolano la dinamica di un neurone i sono:

$$\frac{dy_i}{dt} = \begin{cases} \frac{1}{\tau_i}\left(-y_i + g_i I_i\right) & i \text{ neurone d'ingresso} \\ \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{k} \omega_{ji}\sigma(y_j + \beta_j) + g_i I_i\right) \end{cases} \tag{3.3}$$

Dove y_i rappresenta l'uscita del neurone i-esimo, τ_i la costante temporale, g_i un guadagno, I_i il valore di ingresso del neurone i, ω_{ji} il peso della connessione sinaptica dal neurone j al neurone i, β_j la soglia del neurone j,

$\sigma(y_j + \beta_j)$ l'uscita del neurone j. Tipicamente la funzione σ è una sigmoide: $\sigma(x) = \frac{1}{1+e^{-x}}$.
Diversamente dalle reti reattive descritte in precedenza, quelle ricorrenti sono utilizzate nei compiti nei quali serve che le uscite della rete dipendano non solo dagli input correnti ma anche da eventi precedenti. Ciò consente di affrontare problemi che richiedono memoria, quali ad esempio l'apprendimento[1]. Nel caso si usino queste reti, l'algoritmo evolutivo viene impiegato per individuare i pesi ω_{ji} e gli altri parametri (soglie, guadagni, costanti temporali) in modo che la rete manifesti le dinamiche desiderate.
Un altro modello dotato di memoria sono le reti di tipo plastico (*Plastic Neural Networks, PNN*); in questo caso non è solo lo stato dei neuroni che fa da memoria, ma anche pesi sinaptici che cambiano nel tempo. Questi variano secondo meccanismi analoghi all'apprendimento hebbiano:

$$\omega_{ij}^{t} = \omega_{ij}^{t-1} + \eta \Delta\omega_{ij} \ , \tag{3.4}$$

ω_{ij} è il peso della connessione uscente dal neurone i ed entrante in j, $0.0 < \eta < 1.0$ è il coefficiente di apprendimento e $\Delta\omega_{ij}$ la variazione calcolata secondo regole che tengono conto dei livelli di attivazione dei neuroni pre e post-sinaptici (i e j rispettivamente). Gli indici $t-1$ e t indicano rispettivamente un generico istante temporale ed il successivo.
Nel caso di reti plastiche il compito dell'algoritmo evolutivo è quello di individuare i parametri che definiscano le regole per variare i pesi, ovvero per il calcolo dei $\Delta\omega_{ij}$. I pesi ω_{ij} variano online per apprendimento Hebbiano, applicando formule come la 3.4. Al cambiare dei pesi cambia la relazione ingresso-uscita implementata dalla rete e quindi il comportamento osservato nel momento in cui la rete controlla un robot.
Da notare che con questo tipo di rete il meccanismo che implementa l'apprendimento è predefinito e costituito dalle formule come la 3.4, mentre nelle reti ricorrenti non esiste alcun meccanismo specifico a tale scopo; ciononostante le dinamiche delle *CTRNN* possono portare alla capacità di apprendere.

3.2 Gli algoritmi genetici

Gli algoritmi genetici (*GA*), anch'essi ispirati alla biologia, sono tecniche euristiche usate per affrontare problemi di ottimizzazione per i quali non

[1]Da non confondere l'apprendimento che si ha una volta che la rete viene eseguita sul robot e quello relativo all'evoluzione, che porta ad apprendere parametri di rete nel corso delle generazioni

esistano algoritmi di complessità lineare o polinomiale. L'idea è imitare il processo evolutivo che in migliaia di anni ha portato allo sviluppo delle specie viventi per come le conosciamo oggi, selezionando quelle che più hanno saputo adattarsi alle condizioni imposte dall'ambiente. Alla base della teoria evolutiva c'è il concetto che gli individui più adatti all'ambiente abbiano una maggior probabilità di sopravvivere alle condizioni avverse (clima, malattie, predatori...) e dunque di riprodursi. Con la riproduzione, parte delle caratteristiche che hanno consentito ai genitori di sopravvivere si tramandano alla prole; il procedimento porta ad un graduale adattamento della specie nel tempo.
Gli algoritmi genetici non sono altro che evoluzione artificiale, in cui il criterio di selezione è definito dal programmatore sulla base delle caratteristiche che si vuole abbiano gli individui migliori. L'algoritmo lavora su un insieme di individui, chiamato popolazione, composto da soluzioni codificate ad un problema. Nel caso della robotica evolutiva, il problema è il controllo di un robot che deve svolgere un compito ed ogni individuo è un diverso controllore, potenziale soluzione del problema. Come nel caso biologico anche negli algoritmi genetici si parla di genotipo, il corredo genetico di un individuo e di fenotipo, ovvero di come i caratteri dell'individuo sono espressi. Nell'ambito artificiale, il genotipo rappresenta il modo in cui le informazioni vengono codificate in una stringa di dati: in base alla posizione nella stringa un certo dato assume un significato; il fenotipo è una particolare stringa, un insieme di valori (*geni*) che definisce un individuo (*cromosoma*) e lo distingue dagli altri. I geni (che possono essere numeri, stringhe, istruzioni di un programma...) sono le strutture elementari manipolate dall'algoritmo. Nel caso della robotica evolutiva un cromosoma rappresenta un controllore (tipicamente neurale) ed i geni sono i suoi parametri (pesi sinaptici, soglie di attivazione, costanti temporali...).
L'algoritmo genetico funziona iterando i seguenti passi:

1. *Valutazione* degli individui della generazione corrente;
2. *Selezione* degli individui;
3. *Riproduzione* (creazione della generazione successiva);

La *valutazione* consiste nel verificare la bontà delle soluzioni che costituiscono la popolazione; nel caso dell'evolutionary robotics il controllore viene testato sul robot o in simulazione e si valuta come affronta il compito.
Dopo essere stato valutato, ogni cromosoma riceve un punteggio (dipendente dal modo in cui si è comportato in fase di valutazione) attraverso

una funzione chiamata *fitness function*, la quale incorpora i criteri che rendono un individuo migliore di un altro. Se per esempio si vuole evolvere un controllore per evitare gli ostacoli, allora la fitness function dovrebbe assegnare punteggi più alti ad individui che collidono poche volte. Sulla base del punteggio ottenuto si ha una *selezione* dei cromosomi che contribuiranno a creare la generazione successiva; tipicamente, i cromosomi che hanno ottenuto i punteggi maggiori vengono copiati direttamente nella nuova generazione (elitismo), in questo modo le migliori soluzioni trovate si propagano immutate.
Le modalità più comuni per selezionare gli individui sono: *fitness proportional* (o *roulette wheel*) e *tournament selection*. Nel primo caso, un individuo viene selezionato con un probabilità pari al rapporto tra la sua fitness e il totale della fitness dell'intera popolazione. Il rischio con questo tipo di selezione sta nel fatto che se la fitness di un individuo è molto maggiore di quella degli altri, la frequenza con cui verrà selezionato sarà talmente elevata da impoverire la generazione seguente in termini di varietà del materiale genetico, col rischio di restare intrappolati in ottimi locali. La tournament selection invece consiste nel prendere casualmente n individui dalla popolazione, selezionando poi quello che ha fitness più elevata; questa modalità non soffre del problema messo in evidenza.
Gli individui vengono selezionati per la *riproduzione*, che consiste nella creazione di nuovi cromosomi a partire da quelli della vecchia generazione. La riproduzione consiste nell'applicazione di operatori genetici chiamati *crossover* e *mutazione*; la modalità con cui questi operatori lavorano cambia a seconda del tipo di codifica che si utilizza, ma dal punto di vista semantico ne sono indipendenti. Il crossover opera su coppie di individui, vengono scelti uno o più punti oltre i quali il materiale genetico degli individui viene copiato da un cromosoma all'altro, come mostra la figura 3.4.

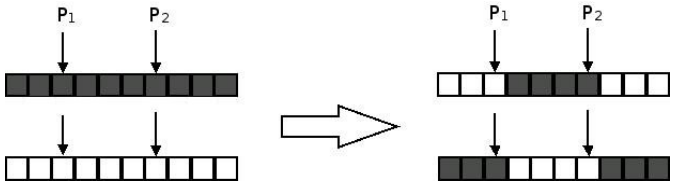


Figura 3.4: Rappresentazione concettuale dell'operatore di crossover in 2 punti (P_1 e P_2).

La mutazione invece viene applicata su un singolo individuo e consiste nell'alterazione dei valori di alcuni geni, selezionati in modo casuale. La modalità di funzionamento dipende dal tipo di codifica che si utilizza: per genotipi

binari la mutazione consiste nell'inversione del valore di alcuni bit; nel caso di codifche con numeri reali, tipicamente viene aggiunto ai geni da mutare un offset campionato da qualche distribuzione (uniforme, normale...). La figura 3.5 mostra i due tipi di mutazione citati.

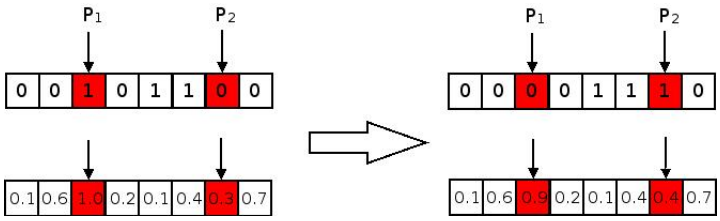


Figura 3.5: Rappresentazione dell'operatore di mutazione in due punti (P_1 e P_2), con geni binari (sopra) e reali.

Una volta prodotta la nuova generazione, gli individui vengono rivalutati e il ciclo ricomincia. Il processo evolutivo può essere arrestato perché il numero di generazioni prodotte raggiunge un valore predefinito, oppure perché la fitness del miglior individuo supera un valore di soglia o non cresce per un certo numero di generazioni.
Tipicamente l'algoritmo viene eseguito per un numero di generazioni che può essere dell'ordine delle migliaia. Questo rappresenta un problema in fase di valutazione degli individui in quanto in genere:

- Il numero di individui di ogni generazione è di svariate decine;
- Ogni individuo viene valutato più volte, in modo da cambiare le condizioni iniziali;
- Ogni singola valutazione può durare diversi minuti.

Da queste considerazioni consegue che il tempo necessario ad evolvere un controllore potrebbe essere dell'ordine di un centinaio di anni. Questo problema viene risolto attraverso l'impiego di simulatori, software che consentono di replicare aspetti del mondo reale e condurre le valutazioni accelerandone i tempi.
La quasi totalità degli esperimenti evolutivi si avvale di ambienti simulati. Nel caso si voglia portare i risultati su robot reali si adottano accorgimenti quali l'impiego di componenti stocastiche all'interno dei meccanismi simulati, per modellizzare l'incertezza nei confronti del mondo reale. Altri approcci sono valutare gli individui nelle ultime generazioni usando il robot vero, oppure nel caso di compiti molto semplici, condurre effettivamente tutte le valutazioni sul robot fisico, come in [15].

Capitolo 4

Metodologie di lavoro

In questo capitolo vengono descritti il problema affrontato e gli strumenti utilizzati per mettere in luce i fenomeni studiati.

4.1 Il robot e-puck

L'*e-puck* [1] è un piccolo robot mobile, nato da un progetto dell' Ecole Polytechnique Federale de Lausanne (EPFL - Losanna, Svizzera). Il robot nasce con l'intenzione di avere un piattaforma che consenta a studenti e ricercatori di effettuare esperimenti in ambiti di robotica autonoma, collettiva e interazione uomo-macchina. In virtù di questi obiettivi le caratteristiche principali del robot sono robustezza meccanica, semplicità d'uso e basso costo.
Il robot è venduto con licenza openhardware: chiunque può produrlo e venderlo, ma non può averne l'esclusiva. Inoltre c'è la libertà di modificare la piattaforma e commercializzare il nuovo prodotto, ma sempre con la medesima licenza e allegando le nuove specifiche in modo che l'acquirente possa potenzialmente riprodurre il nuovo modello o a sua volta cambiarlo.
Caratteristiche hardware del robot:

- 2 motori passo passo movimentano ognuno una ruota, con un rapporto di riduzione 50:1;
- processore dsPIC30F6014A [2] , facilità di programmazione, compatibilità con compilatori GNU;

[1] www.e-puck.org
[2] www.microchip.com

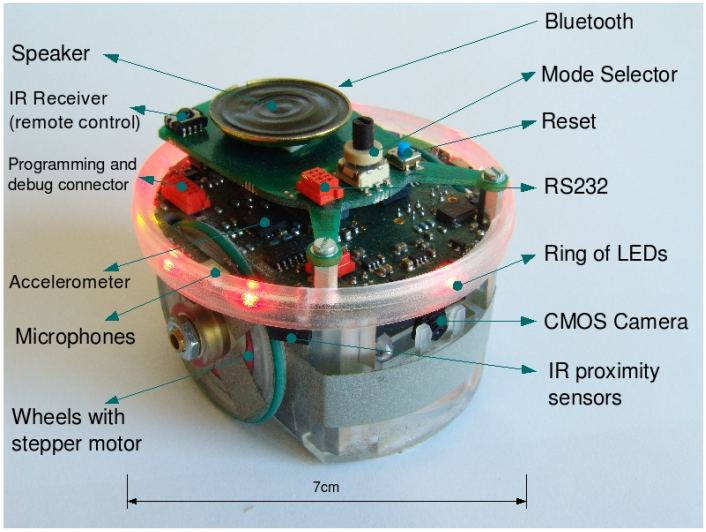


Figura 4.1: robot e-puck

- batteria al litio ricaricabile, 3.4V;
- 3 microfoni, frequenza massima di campionamento 33Khz;
- altoparlante;
- accelerometro 3D;
- 8 sensori di prossimità a infrarossi;
- camera a colori VGA, risoluzione 640*480;
- modulo per connessione bluetooth;

Sono inoltre disponibili le seguenti estensioni:

- Torretta con camera ominidirezionale;
- Ground sensors;
- Anello LED multicolore;

I componenti hardware citati sono gestiti via software attraverso librerie opensource scritte in codice C, reperibili sul sito ufficiale. Ogni libreria fornisce un metodo di inizializzazione del relativo componente ed una serie di funzioni per poterlo utilizzare. Il robot è programmato in codice C: mediante le funzioni di libreria è possibile leggere dati dai sensori e inviare comandi agli attuatori per ottenere il comportamento desiderato. Il codice

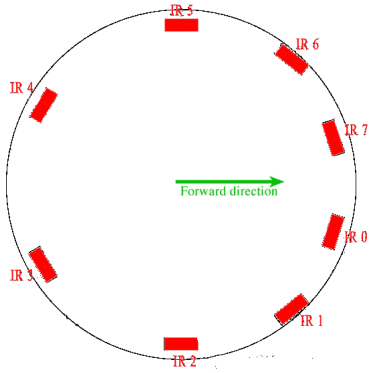


Figura 4.2: Posizione dei sensori a infrarossi sul robot e-puck

viene cross-compilato con i tool make di linux ed il compilatore gcc e può poi essere trasferito sul robot attraverso uno script che si collega in bluetooth e trasferisce il file binario. Il programma viene eseguito una sola volta dal robot, di solito viene implementato attraverso un ciclo infinito che contiene il loop di controllo.
In seguito vengono descritti con maggior dettaglio gli elementi più importanti relativamente al lavoro presentato in questa tesi:

- sensori di prossimità ad infrarossi ed ambient-light;
- microfono;
- agenda.

Gli 8 sensori a infrarossi sono posizionati sull'e-puck come mostrato in figura 4.2[3].
I sensori sono composti da due parti: un emettitore ed un ricevitore. Il primo emette un raggio infrarosso, che viene riflesso dagli oggetti. Parte del segnale torna pertanto verso il sensore e sulla base delle sue caratteristiche viene fornita una misura proporzionale alla distanza dall'oggetto colpito. Nel caso del robot e-puck il valore letto all'interno del programma è la differenza tra il livello di infrarossi nell'ambiente e quello ottenuto dal segnale riflesso. Le letture dei sensori seguono l'andamento mostrato in figura 4.3[4] (la figura rappresenta dati presi da uno specifico robot); in questo caso sono rappresentate le letture dei 4 sensori frontali (IR0, IR1, IR6 e IR7) quando il robot parte a distanza zero da un muro e si muove in modo rettilineo

[3]Immagine presa dal sito www.epuck.org
[4]Immagine presa dal sito www.epuck.org

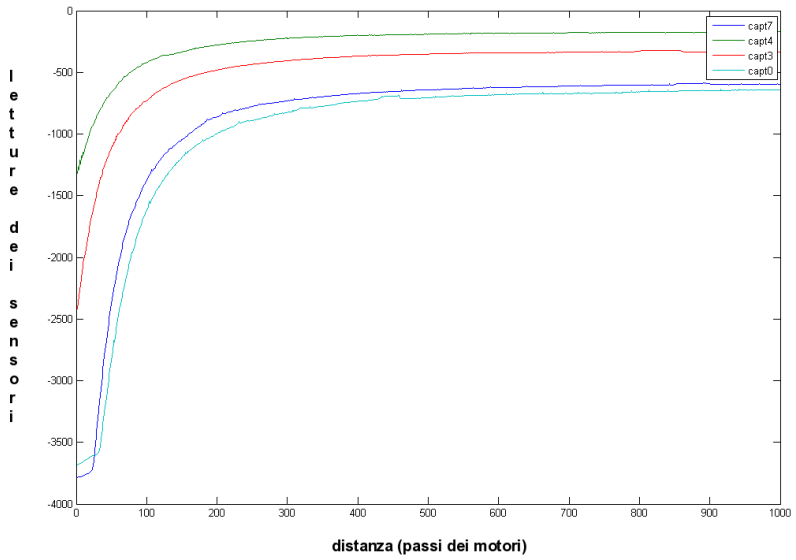


Figura 4.3: Legame tra distanza e lettura dei sensori IR. Dati campionati da un robot specifico.

allontanandosi dal bersaglio. Sulle ascisse sono indicati i passi del motore (1000 passi corrispondono a 12,8 Cm).

Il valore letto dipende, oltre che dalla distanza, da altre caratteristiche quali la forma dell'oggetto colpito, che influenza il modo con cui la luce viene riflessa, ed il colore del bersaglio: gli oggetti chiari riflettono una maggiore quantità di infrarossi e sono individuabili a distanze più grandi rispetto a quelli scuri. In aggiunta, nel caso dell'e-puck, se il robot è in movimento alle letture si aggiunge una componente stocastica dovuta all'attività dei motori, l'entità di questo rumore cresce col diminuire del livello di carica della batteria perchè a seconda del momento ai sensori potrebbe arrivare potenza insufficiente per una misura affidabile.

L'uso tipico dei sensori di prossimità in agenti autonomi è quello di valutare le distanza da oggetti con i quali il robot potrebbe collidere. Sulla base di questa informazione vengono attuate misure di correzione della traiettoria per evitare urti. Un uso meno comune è per la comunicazione, per esempio è possibile definire un codice e inviare segnali binari sincronizzando trasmettitori e ricevitori su robot diversi, oppure i robot possono comunicare attraverso il movimento. In questo caso, ciascuno utilizza i sensori di prossimità per percepire gli altri robot; a seconda di come questi si muovono cambiano gli input sensoriali e quindi i pattern di movimento possono essere

usati per scambiarsi informazioni.
In un esperimento del 2002 Quinn[30] usa i sensori proprio per questo scopo, evolvendo una rete in grado di guidare un gruppo di tre robot in una direzione comune non predefinita, col solo uso di sensori di prossimità. Il comportamento che ne risulta è una sorta di danza preliminare attraverso la quale i robot definiscono dei ruoli per poi muoversi in formazione nella stessa direzione.
L'agenda è un componente software che viene utilizzato come meccanismo di temporizzazione che consente di eseguire operazioni periodiche a intervalli di tempo prefissati. Questo aspetto è molto importante per un controllore neurale in quanto la rete viene evoluta sul simulatore ed i parametri utilizzati nella fase di valutazione dei cromosomi si riflettono sul programma che verrà eseguito sul robot. Uno di questi parametri è la durata del passo di simulazione. Il principio di funzionamento del simulatore prevede un ciclo composto da un certo numero di passi; all'interno di ciascuno vengono simulate le seguenti operazioni:

1. calcolo dei dati sensoriali, in dipendenza dallo stato del mondo;
2. invio dei comandi agli attuatori, da parte del controllore sulla base degli input misurati;
3. attuazione (movimentazione del robot, gestione collisioni, aggiornamento stato del mondo);

Gli effetti del passo 3 dipendono dalla durata del passo temporale, ovvero la durata che si assegna al ciclo descritto. Se per esempio si considera un robot che si muove in moto rettilineo con una velocità di 10 Cm/s, in un passo di durata 0.1 s il robot si sposterà di 1 cm, con un passo di durata 0.2 s 2 cm e così via. Questo parametro influenza quindi sia l'effetto che i comandi dati agli attuatori hanno sul robot e sul mondo sia le caratteristiche degli input sensoriali, quindi l'intero comportamento della rete. A valle di queste considerazioni risulta evidente che, per funzionare in maniera corretta, la rete che viene eseguita sul robot deve lavorare alla stessa frequenza usata in simulazione.
L'agenda implementa un timer cui è associata una funzione di callback, chiamata ad intervalli regolari; codificando il ciclo della rete neurale all'interno di questa funzione e settando come valore per l'intervallo lo stesso usato in simulazione è possibile ottenere un controllore neurale aggiornato alla frequenza desiderata. Intervalli piccoli consentono una maggior precisione sul simulatore e reattività della rete sul robot; ma esistono limiti fisici dovuti alla velocità di calcolo del processore e dei sensori. Inoltre, la durata delle

simulazioni aumenta in modo proporzionale: un passo molto piccolo consente aggiornamenti precisi della fisica del mondo, ma comporta un maggior numero di operazioni a parità di unità di tempo simulata.

4.2 Simulazioni in evolutionary robotics

In evolutionary robotics gli esperimenti vengono condotti quasi sempre in ambienti simulati, sebbene esistano approcci diversi: Floreano e Mondada [15] hanno evoluto un controllore che consentisse ad un robot di esplorare un ambiente evitando gli ostacoli conducendo l'intero esperimento sul robot reale; Nolfi, Miglino e Parisi [27] hanno adottato un approccio ibrido, in cui la parte finale dell'evoluzione era condotta sui robot. Le motivazioni principali per cui la maggioranza dei lavori vengono condotti in simulazione sono due:

1. questo tipo di approccio richiede numerose iterazioni del ciclo evolutivo in cui uno stesso individuo può essere valutato più volte e per periodi medio-lunghi;

2. i contollori neurali nelle prime generazioni producono comportamenti casuali che potrebbero danneggiare il robot.

La valutazione in un ambiente reale risulta il più delle volte impossibile[5], considerando anche che in simulazione non esistono tempi morti per il setup iniziale di ogni valutazione, la gestione di malfunzionamenti, l'effettuazione di misure per valutare i controllori, i cambi delle batterie per i robot...
L'uso di un software consente quindi di affrontare problemi complessi in tempi ragionevoli, eventualmente eseguendo in parallelo più simulazioni per aumentare la probabilità di trovare una buona soluzione.
La difficoltà maggiore nella costruzione di simulatori sta nel fatto che tipicamente si desidera che i controllori che mostrano un certo comportamento in ambiente virtuale ne mostrino uno analogo nel mondo reale. Jakobi [22] pone particolare attenzione a questo aspetto, introducendo il concetto che il comportamento di un agente non dipende solamente dal controllore, ma anche dall'ambiente in cui si trova ad operare. Infatti, gli output degli attuatori definiscono il comportamento corretto solo se hanno determinati effetti sul mondo; allo stesso modo, le letture dei sensori influenzano l'uscita del controllore attraverso la legge di controllo e quindi il comportamento del robot.

[5] Alcuni esperimenti condotti con twodeepuck hanno richiesto la simulazione di circa 800000 ore, più di 90 anni. L'esperimento di Floreano e Mondada una decina di giorni.

Le considerazioni possono sembrare banali, ma in realtà contengono la direttiva principale che dovrebbe guidare nello sviluppo di simulatori: il modo con cui un comportamento si manifesta dipende solamente da alcuni elementi del mondo, questo significa che per avere comportamenti simili sia in simulato che nel reale occorre individuare e modellizzare al meglio tutti e soli gli elementi del mondo che influenzano il comportamento. In ogni caso esisterà sempre una discrepanza tra modello e realtà: non tutti i fenomeni importanti sono colti e modellizzati e comunque nessun singolo fenomeno può essere modellizzato alla perfezione; l'obiettivo diventa quello di ridurre al minimo questa discrepanza, in modo che il comportamento che si ottiene sia soddisfacente anche se non perfettamente uguale. In aggiunta, occorre prestare molta attenzione alla modalità di implementazione dei modelli: potrebbero sorgere caratteristiche dell'ambiente simulato che non trovano riscontro in quello reale e controllori evoluti artificialmente potrebbero sfruttare questi errori per mostrare un comportamento buono in simulazione ma che non è poi possibile replicare nella realtà, dove questi aspetti sono differenti. Nel seguito, viene fornita una panoramica sul software utilizzato per condurre le simulazioni.

4.2.1 Il simulatore twodeepuck

Twodeepuck è il nome del simulatore implementato per svolgere esperimenti evolutivi e facilitare il debugging di programmi da eseguire poi sugli e-puck. Il software è scritto in codice C++ ed è stato sviluppato per lavorare in ambiente linux. L'implementazione è partita dal simulatore *twodee*[10], scritto da Anders Lyhne Christensen per i robot s-bot. La scelta di non implementare un nuovo sistema da zero, oltre che da motivi di comodità, viene dal fatto che gli utenti di twodee potranno svolgere esperimenti coi robot e-puck usando una piattaforma molto simile a quella cui sono abituati. Twodee è stato progettato con l'idea di avere un sistema versatile, che consentisse di creare e personalizzare gli esperimenti in modo rapido; la struttura fortemente modulare dimostra come questo requisito abbia ricoperto un ruolo centrale nella scrittura del codice.
Altra caratteristica ricercata è l'efficienza: gli elementi del mondo simulato e le loro interazioni sono modellizzati in maniera molto essenziale, cercando di catturarne le caratteristiche salienti. I robot per esempio si muovono in un ambiente bidimensionale ed essi stessi sono rappresentati in due dimensioni[6]. Questa scelta deriva dall'obiettivo per cui twodee era stato costruito: evolvere strategie di navigazione di gruppo evitando di cadere nelle buche

[6] da qui il nome del simulatore

presenti nel terreno[7]; questo tipo di mondo è stato mantenuto in virtù dei limiti fisici dell'e-puck che non consentono di operare nella terza dimensione. Per l'implementazione di twodeepuck è stato conservato il nucleo del software precedente cui sono stati aggiunti via via gli elementi necessari per poter simulare i robot e-puck.
Il simulatore è costituito da una serie di classi che rappresentano le diverse entità del mondo; siano esse fisiche, come sensori e attuatori o logiche come ad esempio il gestore delle collisioni. Ad eccezione del render, usato per la visualizzazione, tutte le classi ereditano da una radice chiamata *CSimObject* che definisce i metodi comuni (vedere diagramma di fig.4.4). Caratteristica

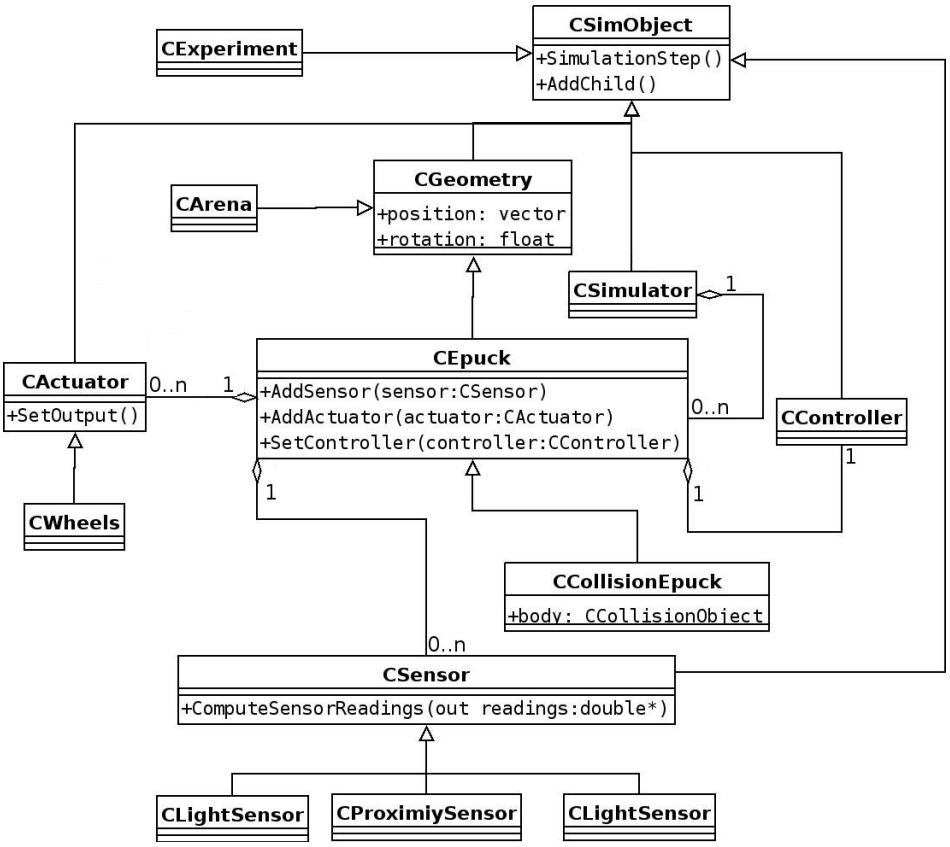


Figura 4.4: Classi principali di twodeepuck

saliente è la possibiltà di creare un albero associando gli oggetti di questa classe tramite relazioni di tipo padre-figlio. Una struttura di questo tipo è motivata dalla modalità di funzionamento del simulatore: esso modellizza un mondo governato da un tempo continuo attraverso una serie di passi dis-

[7]Task che può essere studiato usando un modello a due dimensioni

creti, in ognuno dei quali deve essere cambiato lo stato di tutti gli elementi. Un'organizzazione di tipo gerarchico consente di decentralizzare l'implementazione della logica di aggiornamento in ogni componente, senza aver bisogno di una singola entità che abbia una visione globale del sistema; ciò porta indubbi vantaggi in termini di estendibilità in quanto l'aggiunta di un nuovo elemento non richiede nessun tipo di intervento su quelli già presenti. Il metodo incaricato di gestire questo meccanismo è chiamato *SimulationStep*. Quando in un oggetto si implementa questa funzione la prima cosa da fare ad ogni passo di simulazione è una chiamata al metodo omonimo della classe CSimObject. Dato che per default l'implementazione prevede una chiamata su tutti i figli, a runtime si ha una propagazione lungo l'albero che provoca un aggiornamento dal basso verso l'alto dello stato degli oggetti (i figli aggiornano il proprio stato prima del rispettivo padre).
L'intera simulazione è rappresentata dalla classe *CSimulator* che consiste in una area (arena) in cui si muovono i robot; il suo compito è gestire il flusso temporale e contenere gli oggetti del mondo. Essa, a runtime, fa da radice dell'albero, definendo l'ordine con cui avvengono gli aggiornamenti:

1. Ad ogni passo di simulazione, vengono aggiornate le letture dei sensori per ognuno dei robot presenti;

2. Aggiornate le misure viene propagato il ciclo di simulazione verso i figli, in questo caso il gruppo di robot;

3. A loro volta i robot propagano la chiamata agli oggetti figli, nell'ordine: sensori, attuatori e controllore. Sensori ed attuatori tipicamente non compiono alcuna operazione, il controllore usa le letture dei sensori per cambiare lo stato degli attuatori secondo la logica di controllo;

4. Il flusso del programma torna al robot, che sulla base dello stato in cui si trovano i suoi attuatori modifica i propri parametri, tipicamente la posizione;

5. Infine, una volta che tutti i robot si sono mossi, se è previsto un modello per le collisioni si ha una fase di gestione, finita la quale il ciclo riparte;

Le operazioni descritte sono implementate dal seguente pseudo-codice:

```
\\Simulator SimulationStep
for all r in robots do
        sensors=r->GetSensors()
        for all s in sensors do
```

```
                s->Sense()
        done
done
for all r in robots do
        r->SimulationStep()
done
ManageCollisions()

\\Robot SimulationStep
for all c in children do
        c->SimulationStep()
done
```

Analizzando il ciclo descritto risulta evidente che i sensori sono trattati in maniera diversa rispetto agli altri elementi, il loro stato infatti viene aggiornato in un unico momento: l'albero gerarchico non viene percorso in profondità ma in ampiezza. La ragione è che i sensori devono fornire informazioni sul mondo e con questa strategia si può garantire la coerenza delle misure. Considerando ad esempio due robot vicini e le letture dei rispettivi sensori di prossimità, se il flusso di aggiornamento seguisse l'albero in profondità allora verrebbero calcolate le misure per il primo robot, dopodichè il controllore attuerebbe dei valori che lo farebbero muovere, col risultato che le misure del secondo robot non sarebbero coerenti in quanto la distanza è cambiata. Col metodo descritto, una volta aggiornati i sensori gli altri elementi possono poi essere gestiti in modo sequenziale senza nessun problema. L'aggiornamento dei vari elementi avviene attraverso la chiamata del metodo SimulationStep nei robot, che propaga l'aggiornamento dello stato a tutti gli oggetti figli (controllore e attuatori).
Il simulatore può funzionare in due modalità:

1. esecuzione di un singolo esperimento;
2. esecuzione dell'algoritmo genetico;

Nel primo caso, all'interno del programma principale viene creato l'esperimento ed eseguito una sola volta; nel secondo caso, il controllo passa ad un metodo nel file main che implementa l'algoritmo genetico.

L'algoritmo genetico

Come accennato, l'algoritmo genetico è implementato all'interno di un metodo del file main. I parametri dell'algoritmo (quali ad esempio tasso di mu-

tazione, numero di individui d'elite, durata delle valutazioni ecc.) sono definiti come variabili globali del main file ed alcuni di essi possono essere specificati da linea di comando.
L'algoritmo evolutivo può partire in tre modalità differenti:

1. è possibile inizializzare una popolazione di individui dal nulla, assegnando valori completamente casuali ai geni;

2. si può creare una popolazione casuale partendo da un file che descrive un individuo e alterarne leggermente i valori; in questo modo si ottengono cromosomi molto simili a quello di partenza. Lo scopo è evolvere comportamenti complessi partendo da individui che hanno già alcune delle abilità desiderate, cercando via via di aggiungere le caratteristiche mancanti;

3. è possibile far ripartire delle evoluzioni salvate da un determinato stato; in questo caso la popolazione iniziale viene semplicemente caricata da un file.

Ogni passo dell'algoritmo inizia con la valutazione di tutti gli individui della generazione corrente; tipicamente un individuo viene valutato più volte e si fa la media del punteggio ottenuto, riducendo l'effetto dei componenti aleatori. Terminata la fase di valutazione, si passa alla creazione della generazione seguente. La prima operazione consiste nella copia dei migliori individui (l'elite), dopodichè vengono applicati gli operatori genetici: due individui (padre e madre) sono selezionati dall'insieme di tutti gli individui e con una certa probabilità verranno applicati gli operatori di crossover e mutazione.
Le operazioni che consentono di selezionare gli individui e creare la nuova generazione sono definite all'interno di un metodo della classe *CPopulation* che, come suggerisce il nome, rappresenta una popolazione di individui. Questa scelta consente di diversificare le varie popolazioni in termini di codifica utilizzata, modalità di selezione e nel significato degli operatori di crossover e mutazione, mantenendo la stessa struttura per l'algoritmo genetico. Nel caso degli esperimenti descritti in seguito il crossover viene applicato in un punto generato casualmente, oltre il quale i geni dei due cromosomi vengono copiati da uno all'altro; la mutazione consiste invece nell'aggiungere un offset casuale, campionato da una distribuzione gaussiana, con media nulla e deviazione standard 0,1. Un individuo viene selezionato per la riproduzione con una probabilità pari al rapporto della sua fitness rispetto al totale della popolazione (*fitness proportional selection*).
Nel corso dell'esecuzione dell'algoritmo è possibile salvare dati su file, così

facendo si hanno informazioni utili per analisi a posteriori e si possono poi riutilizzare i controllori evoluti sia in simulazione che sul robot vero.

Robot

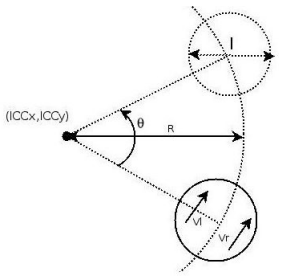
Il robot rappresenta l'elemento più complesso implementato nel mondo simulato. La sua funzione è quella di fare da contenitore per sensori, attuatori e controllore propagando il ciclo di aggiornamento degli stati, e di definire la cinematica per calcolare passo per passo la nuova posizione, data la velocità delle ruote. Viste le dimensioni del robot ed i tipi di task studiati, si è deciso di modellizzare la sola cinematica, trascurando fenomeni dinamici. La scelta è dovuta a ragioni di ottimizzazione: un modello dinamico non avrebbe portato nessun beneficio apprezzabile, ma avrebbe comportato un maggior dispendio di risorse di calcolo ed una minore velocità di esecuzione. Il robot e-puck possiede due ruote che utilizza per la locomozione; pertanto, le equazioni che definiscono la traiettoria percorsa sono quelle del differential drive. Nel caso di un robot che si muova dal punto (X,Y) al punto (X', Y') in un tempo Δ_t:

$R = \frac{1}{2} \cdot \frac{V_l + V_r}{V_r - V_l}$;

$\omega = \frac{V_r - V_l}{l}$

$ICC_x = X - R \cdot sin(\Theta)$

$ICC_y = Y + R \cdot cos(\Theta)$



$X' = (X - ICC_x) \cdot cos(\omega \cdot \Delta_t) - (Y - ICC_y) \cdot sin(\omega \cdot \Delta_t) + ICC_x$

$Y' = (X - ICC_x) \cdot sin(\omega \cdot \Delta_t) + (Y - ICC_y) \times cos(\omega \cdot \Delta_t) + ICC_y$

Le formule includono tutti i casi possibili, eccezion fatta per il moto rettilineo in cui le velocità destra e sinistra sono uguali. In questo caso le formule si semplificano:

$X' = X + V_r \cdot \Delta_t \cdot cos(\Theta)$

$Y' = Y + V_r \cdot \Delta_t \cdot sin(\Theta)$

Controller

Compito di questo elemento è di definire il comportamento che deve tenere il robot quando interagisce col mondo simulato. Ciò richiede che ad ogni passo si leggano le misure dei sensori e sulla base di queste vengano definiti i valori da attuare. La gerarchia delle classi controller è mostrata in figura 4.5.
Un controllore eredita dalla classe *CController* nel cui metodo SimulationStep viene implementato il ciclo di controllo. Il lavoro di tesi presentato coinvolge solamente i controllori neurali; in questo caso, si ha un ulteriore specializzazione nella gerarchia, con la classe *CNNController* che fa da superclasse alle reti neurali e definisce tre metodi astratti:

- *ComputeOutputs*;
- *Reset*;
- *SetWeights*;

Ognuna delle classi figlie si differenzia dalle altre nella modalità con cui questi metodi sono implementati, che dipende dal tipo di rete, mentre l'implementazione del passo di simulazione è nella superclasse ed è comune ad ogni diverso controllore: lettura dei dati dai sensori, chiamata al metodo ComputeOutputs e invio di comandi agli attuatori. Il metodo ComputeOutputs implementa la logica con cui la rete produce le uscite a partire dai valori di ingresso, che si assume siano normalizzati.
Il secondo e terzo metodo della lista permettono ad altre classi di interagire col controllore. Reset è utilizzato dall'algoritmo evolutivo per reinizializzare lo stato del controllore e consiste nell'azzeramento dello stato di attivazione di ogni neurone per reti ricorrenti, mentre per reti reattive (CPerceptronController) il metodo non fa nulla.
Infine il terzo metodo consente di cambiare i parametri della rete, ottenendo di fatto un nuovo controllore ed è utilizzato dall'algoritmo evolutivo nel momento in cui è necessario generare una nuova rete per valutarla. L'input consiste in un array di numeri reali compresi tra 0 ed 1, ognuno dei quali viene poi mappato su un intervallo più grande, a seconda del parametro di rete rappresentato. Per le reti di tipo ricorrente il modo in cui ogni gene viene interpretato può essere cambiato attraverso alcuni flag: per esempio è possibile scegliere se avere la stessa costante temporale per ogni nodo, se essa sia specificata geneticamente o dall'esterno. Così facendo è possibile ridurre la lunghezza del cromosoma e di conseguenza la dimensione dello spazio di ricerca dell'algoritmo genetico.

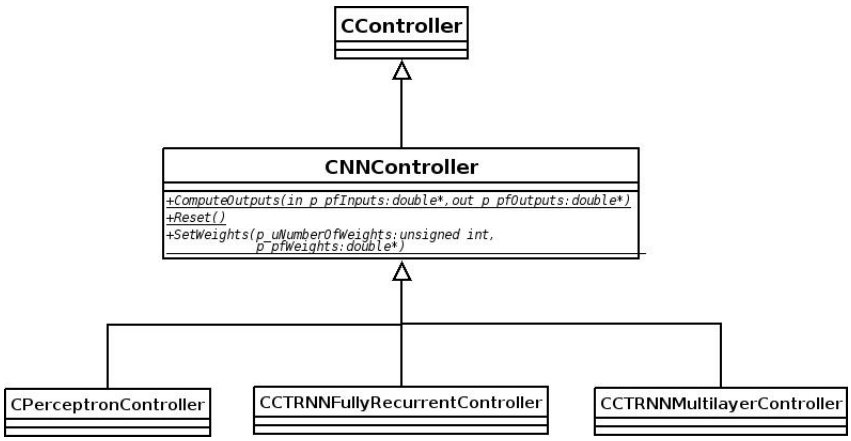


Figura 4.5: Gerarchia della classe CController

Collisioni

Quando i robot si muovono nell'ambiente può accadere che ci siano delle collisioni con altri robot presenti nel mondo o con le pareti; una parte essenziale per un simulatore è quella che si occupa della gestione di questi eventi. Nel caso di twodeepuck al termine di ogni passo di simulazione vi è una fase di gestione degli urti. Gli elementi coinvolti sono:

1. Modello per le collisioni;
2. Collision manager;
3. Collision handler.

Il primo definisce la forma degli oggetti presenti nel mondo influenzando il modo in cui essi collidono; i tipi di geometrie implementati sono rettangoli, cerchi ed anelli rigidi. Tutti gli oggetti sono bidimensionali.
Per modellizzare robot che possono collidere è stata implementata una classe che eredita da CEpuck aggiungendo la definizione della geometria. L'unico modello costruito consiste in una circonferenza di raggio 35mm[8], sufficiente a rappresentare il robot. Ad ogni passo di simulazione questa circonferenza si muove in conseguenza dello spostamento del robot; ciò può provocare delle collisioni. L'individuazione di questi eventi è responsabilità del collision manager. Il meccanismo è molto semplice: poichè la simulazione procede a passi discreti e l'aggiornamento delle posizioni avviene in maniera simultanea[9] può accadere che due o più oggetti vengano posizionati in modo tale

[8] 35mm è il raggio del robot e-puck

[9] Da un punto di vista concettuale, in realtà è sequenziale, ma gli effetti sono gli stessi

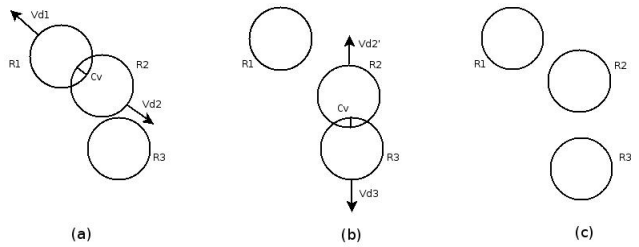


Figura 4.6: Gestione delle collisioni. (a): dopo un passo di simulazione le geometrie R1 e R2 collidono, il collision handler calcola i vettori di riposizionamente V1 e V2, sulla base del vettore Cv calcolato dal collision manager; (b): R1 e R2 vengono riposizionati, ciò causa una nuova collisione con R3, il collision handler interviene nuovamente; (c): i robot R2 e R3 vengono riposizionati.

che si compenetrino l'un con l'altro, ciò costituisce un urto. Il collision manager implementa una serie di metodi che consentono di ottenere informazioni riguardanti questi eventi.
Il collision handler definisce la logica di riposizionamento a seguito di una collisione, riportando il mondo in uno stato consistente. Nel simulatore sono implementate due modalità di gestione degli urti: la più semplice prevede che gli e-puck che hanno urtato qualche oggetto vengano rimessi nella posizione che avevano nel passo precedente. Un altra modalità invece fa si che il robot rimbalzi quando colpisce muri o altri robot; in questo caso la procedura è più realistica ma anche computazionalmente più dispendiosa in quanto richiede un procedimento iterativo per controllare che la nuova posizione assegnata sia valida, ovvero libera da collisioni (vedere fig 4.6).
Nel caso l'ambiente preveda un'arena, anche ad essa si assegna una geometria in modo da modellizzare gli urti con i muri; in twodeepuck è possibile scegliere tra arene circolari e arene rettangolari.
Tutte le proprietà legate alla gestione degli urti possono essere specificate al momento della creazione del simulatore; per esempio è possibile creare un modello di mondo in cui non esistono collisioni; questa flessibilità consente di velocizzare le simulazioni trovando un compromesso con la realisticità.

Gli esperimenti

Gli esperimenti sono lo strumento attraverso il quale è possibile costruire una simulazione, definendo il tipo di oggetti coinvolti. Ogni esperimento eredita dalla classe *CExperiment*, che implementa dei metodi attraverso i quali è possibile aggiungere tutti gli oggetti necessari. Questi metodi sono chiamati in sequenza al momento della creazione del simulatore; per implementare un esperimento basta ereditare da CExperiment, eventualmente

ridefinendo alcuni dei metodi in modo che vengano incontro alle proprie esigenze. L'utente costruisce la propria simulazione scegliendo il tipo di arena e le sue caratteristiche fisiche, i robot e la loro conformazione in termini di sensori, attuatori e controllore. Nel caso si voglia utilizzare l'algoritmo genetico tipicamente vanno reimplementati due ulteriori metodi:

- Reset
- InitializeGeneration

Il primo viene chiamato ogni volta che termina la fase di valutazione di un genotipo ed esegue tutte le operazioni necessarie a reinizializzare lo stato del mondo: tipicamente il riposizionamento dei robot, l'azzeramento del contatore delle collisioni e il reset dei controllori.
Il secondo metodo invece viene utilizzato prima di eseguire tutte le valutazioni degli individui della generazione corrente; il suo scopo è inizializzare dati che devono essere riutilizzati in tutte le valutazioni. Per esempio in certi casi la posizione iniziale in cui si trovano i robot influenza la complessità del compito che devono svolgere, pertanto spesse volte si generano casualmente le posizioni iniziali e le si memorizzano in modo da far partire tutti gli individui nelle medesime condizioni. Ciò riduce il rischio di selezionare degli individui che si sono rivelati migliori di altri solo perché hanno trovato condizioni iniziali favorevoli.

4.3 I robot

In questo capitolo vengono descritti i robot utilizzati per affrontare i compiti descritti nel capitolo seguente; essi sono modelli dell'e-puck, descritto nella sezione 4.1 (pag. 20).
La figura 4.7 mostra la struttura del robot. Dal punto di vista percettivo, il robot è dotato di 8 sensori di prossimità ad infrarossi, 2 sensori di luminosità ed un microfono. I sensori ad infrarossi, IR_i (con i $\in$ [0,7]), sono posizionati a $\pm 17°$, $\pm 50°$, $\pm 90°$ e $\pm 150°$ rispetto alla direzione indicata dalla freccia (fronte del robot). Essi sono modellizzati attraverso una lookup table che, dati distanza e orientamento rispetto ad un oggetto bersaglio, restituisce le 8 letture; i dati contenuti nella tabella sono campioni presi dal robot e-puck. La distanza massima entro la quale è percepibile un ostacolo è di 5 cm. Alle letture viene aggiunto del rumore che varia uniformemente in un intervallo pari al $\pm 20\%$ del valore massimo; il rumore sale al $\pm 30\%$ se nessun ostacolo è nel range. Gli input provenienti dai sensori ad infrarossi vengono utilizzati dalla rete per percepire la presenza di altri robot e mantenere il

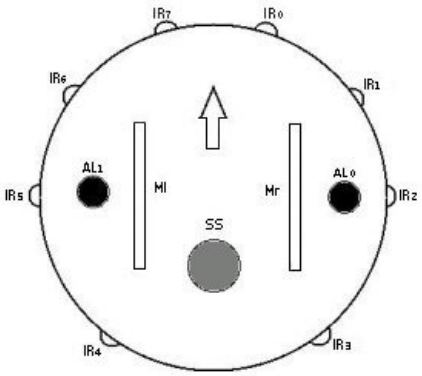


Figura 4.7: Il robot simulato. IR_i, i ∈ [0,7] sono i sensori ad infrarossi; AL_i, i=[0,1] i sensori di luminosità; SS il microfono; Ml il motore sinistro e Mr quello destro.

robot controllato vicino ad essi.
I sensori di luminosità vengono usati per individuare il gradiente luminoso e comportarsi di conseguenza, i sensori sono posizionati a ±90° rispetto alla direzione del robot.
Nelle simulazioni sono stati utilizzati diversi modelli:

1. Modello basato su *lookup table*: le letture dei sensori sono prese da una tabella, contenente dati campionati dal robot e-puck;

2. Modello *lineare*: la lettura di ogni sensore varia linearmente da un minimo di 0, nel caso la luce sia fuori dal raggio massimo, ad un massimo di 1 quando la distanza dalla sorgente luminosa è nulla. Il range è impostato a 1,5 m e l'ampiezza del cono visibile a 60°;

3. Modello *binario* o *a soglia*: l'output del sensore è 1 se la luce è nel campo visivo (1,5 m e 60°), 0 in caso contrario;

Alle letture viene aggiunto del rumore che varia in modo uniforme in un intervallo pari al ±5% del valore massimo (nel caso del sensore a soglia il rumore viene aggiunto nel computo della distanza rispetto alla luce).
I robot sono inoltre equipaggiati con un microfono che consente loro di percepire il suono. Esso è modellizzato in maniera molto semplice: il segnale è privo di sorgente, ovvero è percepibile alla stessa maniera in tutti i punti dell'arena. Il suo valore può essere 0, nel caso in cui non ci sia suono, oppure 1 quando si vuole che i robot lo sentano.
Il fatto di aggiungere il rumore ai dati provenienti dai sensori, non è dato solo dalla necessità di fronteggiare l'incertezza legata alle letture fornite da un singolo sensore, ma sopratutto da quella di coprire le differenze esistenti

tra i sensori di diversi e-puck. Ciò giustifica la scelta di aggiungere il rumore in modo uniforme nonostante il rumore relativo ad un singolo sensore sia, presumibilmente, distribuito in modo gaussiano. Le misure fornite dai sensori, vengono normalizzate nell'intervallo [0,1] e date in input al controllore neurale.

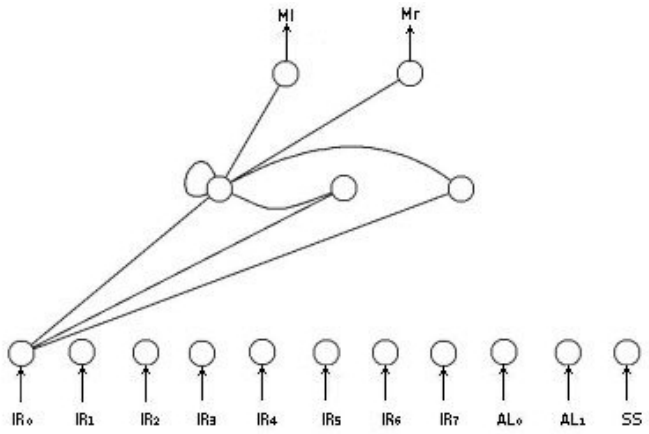


Figura 4.8: La rete neurale che controlla il robot. Per maggior chiarezza sono rappresentate solo le connessioni del primo neurone di ogni strato. Neuroni nello stesso strato sono conessi in modo analogo.

Quest'ultimo è una rete neurale ricorrente, la cui struttura è mostrata in figura 4.8.
Il controllore è dotato di 11 neuroni nello strato di ingresso, 3 neuroni nascosti e 2 d'uscita. I neuroni di ingresso ricevono gli input dagli 8 sensori di prossimità, dai 2 sensori di luminosità e dal microfono. I valori d'attivazione dei neuroni d'uscita vengono mappati nell'intervallo [0,1] attraverso una sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$ e scalati linearmente in [-1000,1000]; il valore ottenuto comanda i motori sinistro e destro. I parametri della rete variano nei seguenti intervalli (vedere sezione 3.1 per il significato dei parametri):

1. Strato di ingresso: la soglia β varia in [-4,-2] ed è la stessa per tutti i neuroni, i pesi sulle connessioni uscenti ω_{ij} sono compresi tra -8 e 8, il guadagno g è lo stesso per tutti i neuroni ed è compreso tra 1 e 13, le costanti di tempo τ_i sono mappate esponenzialmente in $[10^{-1}, 10^{1.6}]$ in cui il valore minimo corrisponde alla durata del passo di integrazione usato per aggiornare la rete ed il valore massimo a circa 40 secondi;

2. Strati nascosto e d'uscita: la soglia β varia in [-5,5] ed in questo caso cambia da neurone a neurone (nel caso dei neuroni d'uscita la soglia

è 0), i pesi sulle connessioni uscenti ω_{ij} sono compresi tra -10 e 10, il guadagno g è 1, le costanti di tempo τ_i sono mappate esponenzialmente in $[10^{-1}, 10^{1.6}]$;

Questa rete viene rappresentata con 69 valori reali compresi tra 0 ed 1, manipolati dall'algoritmo genetico. Di questi valori, 11 rappresentano le costanti temporali dei neuroni di ingresso, 1 la soglia di ingresso, 1 il guadagno; 6 rappresentano soglie e costanti temporali nello strato nascosto, 2 le costanti temporali di uscita ed i rimanenti 48 i pesi delle connessioni. Ognuno di questi valori viene poi mappato linearmente sul rispettivo intervallo, esponenzialmente per le costanti temporali.

Capitolo 5

Gli esperimenti

Questo capitolo descrive i compiti presi in esame per studiare i comportamenti di apprendimento individuale e sociale e le strategie evolutive adottate per ottenere il controllore neurale in grado di affrontare i due compiti. Alcune definizioni sono necessarie per comprendere al meglio quanto esposto: il termine *fototassi* indica un comportamento per il quale si ha movimento verso una sorgente luminosa. *Antifototassi* indica quello opposto, ovvero l'allontanarsi da sorgenti luminose. Nelle sezioni che seguono vengono descritti il compito di apprendimento individuale e quello di apprendimento sociale; segue una descrizione dettagliata degli esperimenti evolutivi che hanno portato all'evoluzione di due reti neurali, in grado di svolgere una il solo compito di apprendimento individuale e l'altra entrambi i compiti.

5.1 Il compito di apprendimento individuale

Il compito studiato per valutare le capacità di apprendimento individuale è ispirato al lavoro di Di Paolo [13], cui sono state apportate alcune modifiche. L'esperimento viene eseguito in un mondo di dimensioni illimitate, privo di ostacoli, in cui c'è una sorgente luminosa. Un robot viene posizionato in questo spazio, la luce è ad una distanza casuale, compresa tra i 55 ed i 65 cm e con un orientamento casuale rispetto al robot. Al robot è richiesto di effettuare fototassi, utilizzando i dati provenienti dai sensori di luminosità. Ogni volta che il robot raggiunge la luce, ovvero è a meno di 5 cm di distanza, viene rimesso nella posizione iniziale e la luce spostata in una nuova posizione.
Il procedimento si ripete fino al momento in cui il robot percepisce un suono. Il suono può essere emesso quando la distanza del robot dalla luce è metà

di quella iniziale; il suono dura dagli 1,5 ai 2,5 secondi.
Dalla percezione del suono in poi il robot deve effettuare antifototassi. Ogni volta esso si allontana dalla luce, ovvero raggiunge una distanza maggiore di 1,5 volte quella iniziale, il robot viene riposizionato e la luce spostata. Questo comportamento (antifototassi) deve continuare fino al momento in cui la rete che controlla il robot viene resettata: lo stato dei neuroni viene impostato a zero e quanto appreso dalla rete viene cancellato, causando il ritorno ad un comportamento di fototassi.

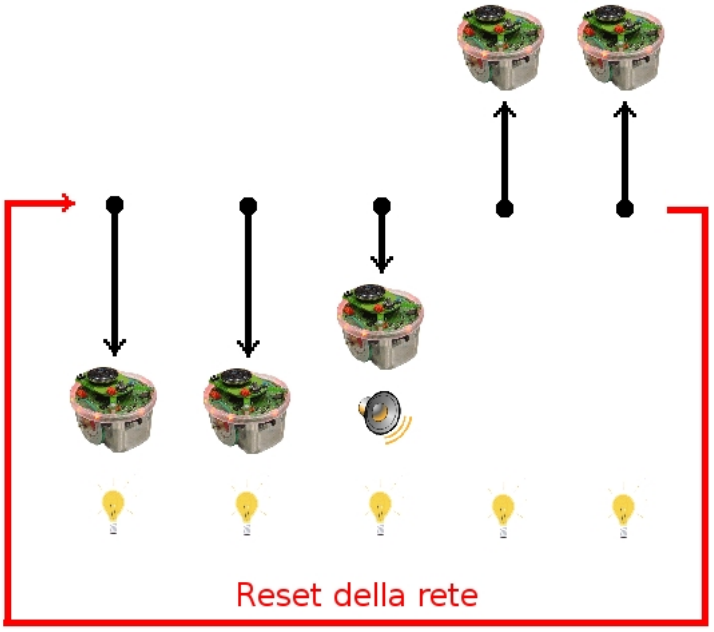


Figura 5.1: Sequenza di 5 trial di learning individuale. Il robot effettua fototassi per 2 trial, nel terzo percepisce il suono, nei 2 trial seguenti effettua antifototassi. La rete viene poi resettata: quanto appreso viene cancellato e il robot ricomincia ad avvicinarsi alla luce.

La figura 5.1 mostra schematicamente un esempio quanto descritto: il robot effettua 5 prove, nelle prime 2 raggiunge la luce (fototassi), ogni volta viene riposizionato. Nella terza prova percepisce il suono, ne consegue che nelle seguenti 2 il comportamento è di antifototassi. Infine, al termine della quinta prova, la rete viene resettata: quanto appreso viene cancellato e il robot riprende il comportamento di fototassi, in attesa che il suono venga nuovamente percepito.
Questo esperimento viene utilizzato per verificare la capacità di apprendi-

mento individuale, ovvero attraverso l'interazione con l'ambiente: si ha apprendimento quando la rete è in grado di associare il corretto movimento al fatto di aver ricevuto o meno l'input sonoro. Il compito è lo stesso descritto da Di Paolo nel suo esperimento, ciò che cambia sono i parametri caratteristici (distanza dalla luce e durata del suono) ed il fatto che Di Paolo usa reti plastiche per controllare il robot, mentre in questo lavoro le reti sono ricorrenti.

5.2 Il compito di apprendimento sociale

Il compito di apprendimento sociale mira a verificare le capacità di apprendere attraverso l'interazione tra robot. Il mondo è lo stesso utilizzato per l'apprendimento individuale, ovvero un arena illimitata in cui c'è una luce, ma in questo caso il numero di individui varia da 1 a 2. Si distinguono due ruoli:

1. Il *demonstrator* è un robot "esperto", cioè che conosce a priori il tipo di movimento da fare nei confronti della luce (fototassi o antifototassi). Sostanzialmente il demonstrator è un robot in grado di svolgere il compito di apprendimento individuale descritto nella sezione precedente;

2. Il *learner* è un robot che deve interagire col demonstrator, cercando di capire come questo si stia muovendo rispetto alla luce e apprendendo lo stesso tipo di comportamento, tramite osservazione. Il learner è inoltre in grado, come il demonstrator, di svolgere il compito di apprendimento individuale.

La figura 5.2 mostra la sequenza che rappresenta il compito di apprendimento sociale.
I due robot sono controllati da reti neurali che hanno la stessa struttura topologica (vedere sez. 4.3), ma diversi valori per i parametri di rete.
Il compito si divide in due fasi distinte. Inizialmente entrambi i robot sono posizionati nel mondo (figure 5.2a e 5.2e). La luce è posizionata casualmente ad una distanza compresa tra 55 e 65 cm dal centroide della coppia. Quando la prova comincia il demonstrator (robot chiaro nelle figure) inizierà a muoversi, avvicinandosi (figura 5.2b) o allontanandosi (5.2f) dalla luce. Compito del learner è sfruttare le misure provenienti dai sensori ad infrarossi per percepire il movimento del demonstrator e cercare di seguirlo. Quando il demonstrator raggiunge la luce (arriva a meno di 5 cm) oppure

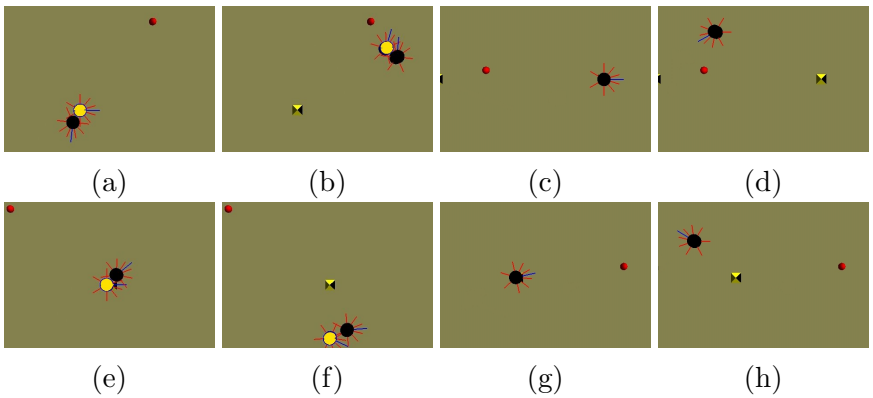


Figura 5.2: Sequenza di eventi che caratterizza i trial di social learning. In alto il caso in cui i robot effettuano fototassi. Sotto il caso in cui i due effettuano antifototassi. La luce è rappresentata dalla sfera, il quadrato è il centro dell'arena.

si è allontanato sufficientemente (supera una distanza di 1,5 volte quella iniziale), la prima parte dell'esperimento termina ed inizia la seconda. Questa prevede che il demonstrator venga rimosso dal mondo, il learner posizionato al centro e la luce spostata (figure 5.2c e 5.2g). A questo punto il learner deve ripetere il comportamento osservato nel demonstrator, avvicinandosi o allontanandosi dalla luce (figure 5.2d e 5.2h rispettivamente).
L'esperimento consente di verificare le capacità di apprendimento sociale: si ha apprendimento se il comportamento osservato nel learner nella seconda fase è lo stesso del demonstrator nella prima.
L'apprendimento è di tipo sociale in quanto, mentre nell'altro esperimento l'evento scatenante l'apprendimento è la percezione del suono, in questo caso, poichè il suono non viene mai percepito, l'unico modo che il learner ha di capire come muoversi è imitare il demonstrator. Il compito di apprendimento sociale è, da un punto di vista evolutivo, più ostico in quanto, mentre nell'apprendimento individuale esiste un input ben definito (il suono) che indichi di cambiare il comportamento, nell'altro caso è il processo evolutivo che deve individuare quali input determinino il comportamento da tenere.

5.3 Esperimenti evolutivi

Obiettivo degli esperimenti evolutivi è evolvere una rete in grado di affrontare i due compiti descritti, ovvero di apprendere sia individualmente tramite il suono, che socialmente, attraverso l'osservazione di un altro robot. Per ottenere questo risultato sono state prese in esame strategie alternative. L'idea iniziale era di avere la stessa rete sui due robot (learner e demonstra-

tor). Data la difficoltà del compito, in termini di capacità che la rete deve saper mostrare, si è invece deciso di usare controllori diversi per i due robot. La differenza non è topologica, ma nel valore dei parametri: la struttura dei due controllori, in termini di numero di nodi e loro connessioni, è comune ai due robot, ma cambiano i pesi ed i parametri relativi ai nodi (soglie, costanti di tempo, guadagni); di conseguenza cambia il tipo di comportamento che la rete manifesta.
A valle di questa scelta restava aperta un'altra questione che riguarda invece la modalità con cui ottenere i due controllori. Le due alternative erano di evolvere contemporaneamente due popolazioni oppure di dividere il processo in due fasi, ottenendo prima una e poi l'altra rete. L'idea che sembrava migliore era la prima, perché coevolvere due individui avrebbe portato probabilmente a maggior sinergia nei ruoli e forse migliori performance. Tuttavia pareva molto più complessa, in quanto il processo evolutivo sarebbe risultato più lento nel momento di valutare gli individui: sarebbe stato necessario avere due diverse popolazioni (una di learner e una di demonstator) e testare le prestazioni di ogni possibile coppia di individui provenienti ognuno da una delle popolazioni.
Si è deciso quindi di avere due reti separate, evolute in due diversi momenti in cui il primo prevedeva di ottenere il demonstrator, il secondo l'altro robot. Il rischio era di complicare ulteriormente il compito col fatto che il learner avrebbe dovuto seguire un demonstrator pre-evoluto: la cosa é piú complessa rispetto al caso in cui il demonstrator evolve assieme al learner in quanto in quest'ultimo caso si svilupperebbero strategie di mutua coordinazione, che vengono a mancare quando una delle reti ha un comportamento fisso (il demonstrator) cui l'altra si deve adattare nel corso dell'evoluzione.
Alcune definizioni sono necessarie per comprendere al meglio i contenuti delle sezioni seguenti:

- Il termine *trial* indica una generica prova, che inizia quando i robot sono al centro dell'arena (solo il learner se la prova prevede un solo robot) e termina nel momento in cui i robot si sono avvicinati (meno di 5 cm dalla luce) o allontanati (più di 1,5 volte la distanza iniziale) rispetto alla luce. Nel caso di apprendimento sociale nella fase in cui i robot sono due, un trial termina quando il demonstrator si è avvicinato o allontanato dalla luce.

- Il termine *valutazione*, riferito ad un controllore, indica l'esecuzione di un trial e assegnazione di un valore di fitness.

- Il termine *generazione* è preso dalla terminologia degli algoritmi evo-

lutivi e rappresenta l'insieme di individui generati ad ogni passo dell'algoritmo genetico, attraverso la selezione e riproduzione (vedere sez. 3.2, pagina 14).

- Il termine *evoluzione* indica l'esecuzione di un'istanza dell'algoritmo genetico, dalla creazione della prima generazione, fino al termine. Tipicamente, per ogni esperimento sono state condotte più evoluzioni, che si differenziano le une dalle altre per l'inizializzazione del generatore di numeri casuali. In questo modo si hanno maggiori possibilità di ottenere buone soluzioni in diverse evoluzioni.
- Il termine *reset*, riferito ai controllori neurali, indica il fatto che gli stati dei neuroni vengano azzerati, cancellando la memoria di quanto appreso.

La figura 5.3 mostra come le varie definizioni sono legate fra loro.

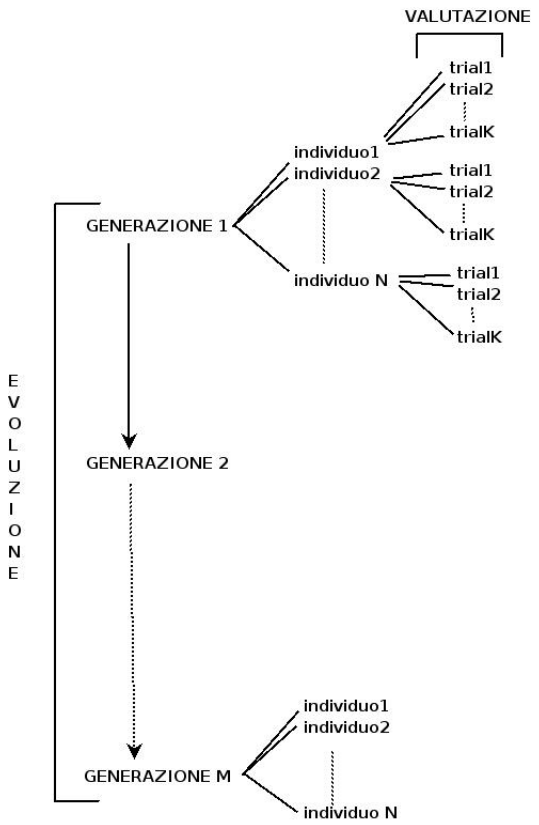


Figura 5.3: Funzionamento dell'algoritmo genetico fino alla generazione M.

L'esecuzione dell'algoritmo genetico (*evoluzione*) porta alla creazione di M *generazioni*. Ogni generazione comprende una popolazione di N *individui*,

creata a partire dalla precedente attraverso l'applicazione di elitismo, crossover e mutazione (vedere sezione 3.2, pagina 14 per maggiori dettagli). Ogni individuo di ogni generazione viene testato per K *trial*, che portano all'assegnazione di un punteggio di fitness all'individuo (media sui K trial), usato poi per la selezione.
Gli esperimenti evolutivi, che hanno portato all'evoluzione di una rete in grado di affrontare entrambi i compiti, sono stati condotti dapprima utilizzando sensori di luminosità a caratteristica lineare e in un secondo momento sono stati rieseguiti, con qualche modifica, usando sensori a caratteristica binaria. Questo perchè nel primo caso si erano ottenuti risultati buoni, ma che presentavano alcuni problemi, risolti poi con variazioni del setup sperimentale (vedere in seguito per maggiori dettagli). Le sezioni che seguono sono tuttavia suddivise rispetto ai due ruoli coinvolti (learner e demonstrator) e non rispetto all'ordine cronologico con cui gli esperimenti sono stati eseguiti, in modo da raggruppare aspetti piú strettamente correlati. La sezione seguente riguarda l'evoluzione del demonstrator e gli esperimenti eseguiti a tal scopo; la sezione 5.5 descrive quanto effettuato per evolvere la rete del learner e comprende l'analisi delle prestazioni nei due compiti.

5.4 Evoluzione del demonstrator

Come detto la strategia evolutiva scelta prevede l'evoluzione di due reti diverse per controllare i due robot. Questa sezione descrive gli esperimenti evolutivi condotti per ottenere il controllore del demonstrator, da usare poi nell'apprendimento sociale. Il demonstrator è, in sostanza, un robot in grado di affrontare il compito di apprendimento individuale descritto nella sezione 5.1 e una volta appreso, fare da guida per il learner nel compito di apprendimento sociale. La rete che controlla il demonstrator è stata evoluta utilizzando una variante dell'esperimento di apprendimento individuale: il numero di individui che eseguono il compito varia da un minimo di 1 ad un massimo di 3. Questa scelta è motivata dal fatto che il controllore che si vuole evolvere verrà poi utilizzato nell'esperimento di apprendimento sociale come guida per gli altri robot, pertanto deve essere in grado di svolgere il compito anche nel caso siano presenti più individui. Questo aspetto non era stato preso in considerazione in un primo momento ed i controllori che si ottenevano erano in grado di svolgere il compito quando soli, ma nel momento in cui altri elementi venivano aggiunti perdevano questa capacità e pertanto non sarebbero stati utilizzabili come demonstrator. Questo avveniva perché, in fase evolutiva, gli input provenienti dai sensori di prossimità ad

infrarossi erano rimasti sempre inattivi, pertanto la rete non era poi in grado di gestire ingressi diversi nel momento in cui altri robot erano percepiti. Di conseguenza, il comportamento che ne risultava era errato.
Inizialmente i robot vengono posizionati in modo che il centroide del gruppo sia in centro all'arena, ognuno ad 1 cm dagli altri (vedere fig. 5.4a); ogni robot ha un orientamento casuale rispetto alla sorgente luminosa (nell'angolo in basso a sinistra nelle figure), la quale viene posizionata casualmente ad una distanza compresa tra i 55 e i 65 cm dal centro.

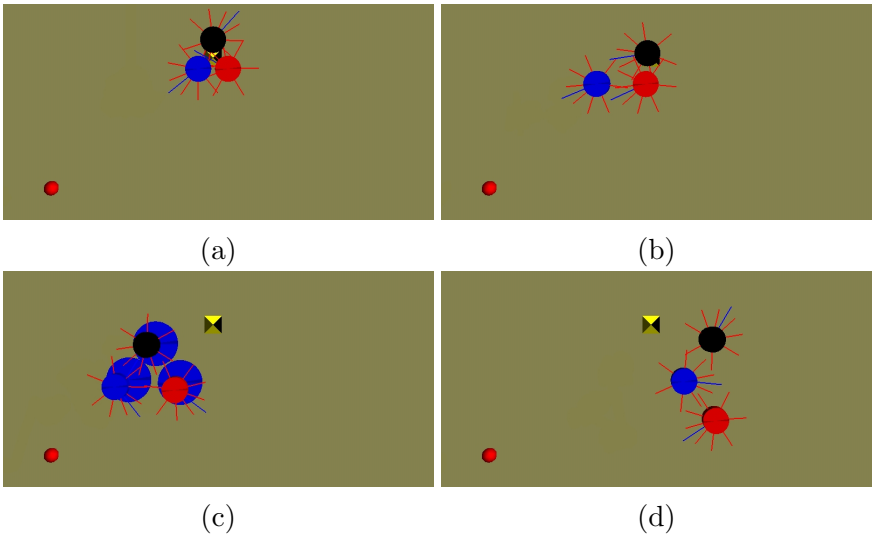


Figura 5.4: Rappresentazione della sequenza che caratterizza un trial in cui viene emesso il suono, la sfera è la luce, il quadrato è in centro all'arena. (a) Situazione iniziale, con i robot al centro dell'arena. (b) I robot iniziano a muoversi verso la luce. (c) Il suono viene percepito (rappresentato dal cerchio attorno ai robot). (d) Il gruppo si allontana dalla luce.

I robot devono seguire la sorgente luminosa (fig. 5.4b); al termine di ogni trial il gruppo viene riposizionato al centro dell'arena e la luce viene spostata in un nuovo punto. I trial sono organizzati in gruppi: ogni 8 il controllore viene resettato; così facendo si possono bilanciare il numero di trial in cui i robot devono fare fototassi e quello in cui i robot devono effettuare antifototassi. In uno (e uno solo) degli 8 trial viene emesso un segnale sonoro: quando la distanza tra il centroide del gruppo e la luce raggiunge metà del valore iniziale i robot percepiscono un suono (fig. 5.4c) la cui durata varia dagli 1,5 ai 2,5 secondi. Lo scopo del segnale sonoro è di indicare ai robot che è il momento di cambiare modo di muoversi rispetto alla luce. Da quel punto in avanti il gruppo deve allontanarsi dalla sorgente luminosa (fig. 5.4d). Ogni volta che il centroide del gruppo raggiunge una distanza pari ad una volta e

mezza quella iniziale (ovvero il trial termina) i robot vengono riposizionati al centro dell'arena, la luce cambia posizione ed i robot devono continuare ad allontanarvisi finchè il controllore non viene resettato.
L'apprendimento individuale si manifesta nel fatto che i robot si allontanino dalle sorgenti luminose una volta che il suono è stato prodotto. Si considera che i robot si siano allontanati solo nel caso in cui la distanza dalla luce superi 1,5 volte quella iniziale. Anche in questo caso i trial sono divisi in gruppi, come illustrato nella figura 5.1, con la differenza che non si ha un solo robot, ma un gruppo.
Prima di affrontare il problema dell'evoluzione del demonstrator, utilizzando l'esperimento appena descritto, sono stati condotti altri esperimenti preliminari il cui scopo era di verificare le potenzialità di apprendimento della rete e individuare una morfologia per il robot che fosse adatta ad affrontare i compiti studiati. La sezione che segue descrive questi primi esperimenti.

5.4.1 Esperimenti preliminari

Inizialmente, il lavoro ha visto una fase di sperimentazione con lo scopo di individuare una morfologia per il robot che gli consentisse di risolvere il problema di apprendimento individuale. Gli aspetti più critici sono i sensori di luminosità ed il controllore. I primi, combinati con le caratteristiche della luce, definiscono la facilità con cui il robot può seguire il gradiente luminoso. Controllori diversi hanno invece diverse potenzialità; dato che il task richiede memoria si è pensato di usare reti ricorrenti, cercando di avere un controllore che fosse il più semplice possibile.
In un primo esperimento, il robot è dotato di tre sensori di luminosità con caratteristica distanza-misura di tipo lineare, posizionati a $\pm 40°$ e $180°$, di raggio 4 metri e ampiezza di $30°$. La scelta di usare un sensore posteriore deriva dal fatto che inizialmente si pensava potesse essere utile al robot per effettuare antifototassi. Il robot non è equipaggiato con i sensori di prossimità ad infrarossi ed il numero di robot è stato ridotto a uno, in modo da avere il caso piú semplice possibile, che consentisse comunque di verificare le capacitá di apprendimento della rete. Il fatto di usare un set di sensori diverso si ripercuote anche sul controllore, che è pertanto differente da quello descritto nella sezione 4.3: il numero di nodi nello strato di ingresso è 4 invece di 11. Le altre caratteristiche del robot sono invariate rispetto a quanto descritto.
Di Paolo nel suo esperimento illustra le difficoltà incontrate nell'evolvere un controllore che prestasse la dovuta attenzione alla luce, proponendo due strategie alternative per affrontare il problema. La strategia scelta prevede

di pre-evolvere gli individui per alcune generazioni affinchè effettuino fototassi, per poi passare all'esperimento completo. Ciò va fatto cercando di evitare di raggiungere livelli di fitness troppo elevati, altrimenti il rischio è che i robot perseverino in questo comportamento anche quando viene introdotto il microfono e gli individui dovrebbero sviluppare un comportamento (antifototassi) che consideri anche l'input sonoro.
Una popolazione di 100 cromosomi, composti da 45 geni[1] inizializzati casualmente tra 0 ed 1, è stata evoluta per un massimo[2] di 2000 generazioni, il numero di individui d'elite fissato a 5, la probabilità di crossover al 35% e quella di mutazione al 6%. Ogni individuo viene valutato per 80 trial, con un limite massimo di 20 secondi per ognuno, scaduti i quali la prova viene fermata. I parametri caratteristici dell'esperimento sono diversi da quanto descritto nella sezione precedente: la distanza della luce dal centro varia dai 45 ai 70 cm, la durata del suono da 0,5 a 1,5 secondi. Il motivo per cui si è usato un setup diverso è il fatto che questi esperimenti avevano carattere esplorativo.
Per evitare di favorire alcuni individui rispetto ad altri, le posizioni della luce e le durate del suono, una volta generate casualmente all'inizio di ogni generazione per ognuno dei trial, sono salvate e riutilizzate per tutti i cromosomi. I trial sono organizzati in 4 gruppi di 10 (reset ogni 10 trial), col suono prodotto al 4°, 5°, 6° o 7° (vedere anche la figura 5.1, pagina 40). Così facendo si hanno 8 trial col suono, 36 in cui i robot devono avvicinarsi alla luce e 36 in cui devono allontanarvisi (vedere tabella 5.1 per maggiori dettagli).

Tabella 5.1: Suddivisione dei trial. G_i indica l'i-esimo gruppo di 10 trials, T_i l'i-esimo trial nel gruppo, P un trial in cui il robot deve effettuare fototassi, S il trial in cui il suono viene percepito e A un trial in cui il robot deve fare antifototassi.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
G_1-G_2	P	P	P	**S**	A	A	A	A	A	A
G_3-G_4	P	P	P	P	**S**	A	A	A	A	A
G_5-G_6	P	P	P	P	P	**S**	A	A	A	A
G_7-G_8	P	P	P	P	P	P	**S**	A	A	A

[1] L'uso di un set di sensori più piccolo influenza la morfologia del controllore e dunque la sua codifica

[2] Spesso gli esperimenti venivano fermati perché si ottenevano buoni controllori prima del limite

Al termine di ogni valutazione un individuo riceve un punteggio calcolato con la formula:

$$F = \begin{cases} \frac{d_i - d_f}{d_i - 5} & \text{nei trial di fototassi} \\ \frac{d_f - d_i}{0.5 \cdot d_i} & \text{nei trial di antifototassi} \\ 0 & \text{se nel trial viene emesso il suono} \end{cases} \tag{5.1}$$

dove d_i è il valore iniziale della distanza tra il robot e la luce espresso in centimetri, d_f quello finale. La costante 5 rappresenta la distanza tra robot e luce sotto la quale un trial viene fermato in quanto si assume il robot abbia fatto fototassi. 0.5 volte d_i è invece il valore da aggiungere alla distanza iniziale per individuare la soglia sopra la quale si assume il robot abbia effettuato antifototassi. Ogni volta che il robot supera una delle soglie indicate il trial termina, così come nel caso il valore della fitness F decresca per più di 50 passi consecutivi. Questa ultima condizione è importante per far sì che il robot presti attenzione alla luce quando deve allontanarvisi: in esperimenti precedenti questo aspetto non era stato considerato col risultato che emergevano strategie che portavano i robot, al momento di effettuare antifototassi, ad andare sempre a velocità massima in una direzione prefissata indipendentemente dalla posizione della luce, confidando sulla bassa probabilità con cui tale strategia li avrebbe portati a meno di 5 cm dalla sorgente luminosa. In questo modo nei 20 secondi di tempo limite i robot riuscivano, pur passando molto vicino alla luce ad allontanarvisi dalla parte opposta, ricevendo un buon punteggio. Un individuo riceve una fitness finale pari alla media dei punteggi ottenuti negli 80 trial.
Nella maggior parte delle evoluzioni, che si diversificano in termini di inizializzazione del generatore di numeri casuali, in meno di 1000 generazioni viene trovata una soluzione. La figura 5.5 mostra l'andamento della fitness in una delle evoluzioni.
Il grafico mostra chiaramente che i migliori individui già attorno alla centesima generazione sono in grado di svolgere il compito in modo corretto, avvicinandosi al valore massimo di 0.9[3]. Il calo drastico del valore della fitness vicino alla generazione 10 è dovuto alla strategia utilizzata che, come detto, prevede che inizialmente i controllori vengano valutati sulla sola fototassi, per poi passare al compito completo una volta che le performance del migliore avessero superato una certa soglia (0.75 nell'evoluzione presa in esame).
Scopo dell'esperimento non era tanto l'ottenere un controllore utilizzabile

[3]I trial con suono danno punteggio nullo, gli altri al massimo 1.

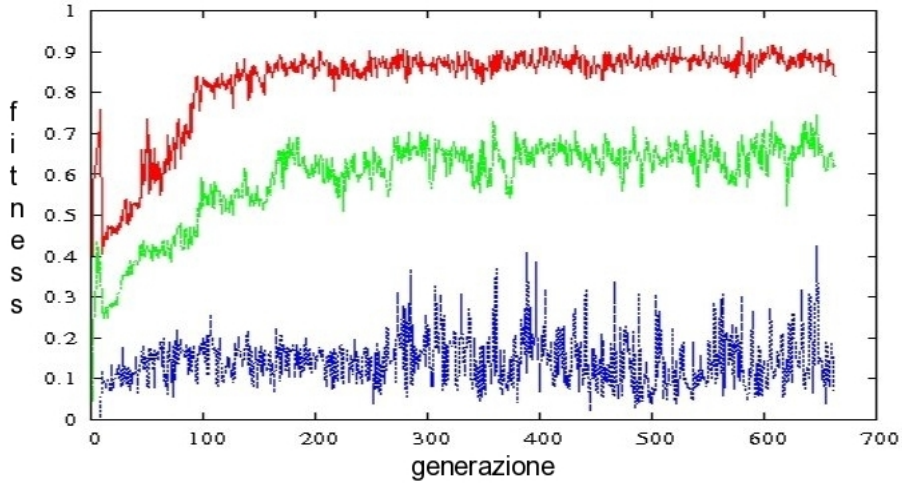


Figura 5.5: Andamento della fitness F, il grafico riporta i valori massimo, medio e minimo di ogni generazione.

come demonstrator, quanto di analizzare le potenzialità del tipo di rete utilizzato; anche a valle delle conclusioni cui giunge Di Paolo nel suo lavoro, in cui afferma che reti di tipo plastico hanno mostrato performance migliori delle altre. I risultati sperimentali indicano invece che l'uso di reti ricorrenti può bastare ad affrontare il problema, almeno per quanto riguarda l'apprendimento individuale; risultato non inaspettato, visti i risultati di altri lavori ([33, 34, 29]).
Una volta verificato che la rete fosse in grado di svolgere il compito di apprendimento individuale si è passati alla fase successiva, il cui scopo è l'evoluzione della rete del demonstrator. Le due sezioni che seguono descrivono gli esperimenti evolutivi che hanno portato ad ottenere diverse reti utilizzabili per controllare il demonstrator.

5.4.2 Evoluzione del demonstrator con sensori di luminosità a lookup table

Dati i risultati positivi ottenuti con quanto descritto nella sezione precedente l'esperimento è stato poi rieseguito, aggiungendo al robot i sensori ad infrarossi, in modo che potesse percepire la presenza di altri. Dato che lo scopo dell'esperimento era di ottenere il demonstrator il compito ha coinvolto più robot, secondo quanto descritto in 5.4 (pag. 45). Se non si fosse adottata questa strategia non ci sarebbe stata alcuna pressione selettiva nei confronti dei parametri relativi ai nodi connessi ai sensori ad infrarossi. Ciò vuol dire che in sostanza tali parametri avrebbero avuto valori del tutto ca-

suali, dunque il comportamento dei robot sarebbe stato errato nel momento in cui fosse stata percepita la presenza di altri individui.
I robot vengono posizionati in modo che il centroide del gruppo corrisponda con l'origine del sistema di coordinate. I robot sono ad 1 cm di distanza l'uno dall'altro e con orientamento casuale. La distanza della luce e la durata del suono sono le stesse usate nel caso precedente, il numero di trial passa invece da 80 a 120 in modo da avere 40 trial (18 di fototassi, 18 di antifototassi e 4 col suono) per ogni dimensione del gruppo di robot. Un'altra differenza riguarda i robot: anche il modello dei sensori di luminosità, come per gli infrarossi, integra dati provenienti da una procedura di campionamento sul robot vero. La scelta derivava dall'intenzione di portare eventuali risultati positivi sull'e-puck.
Data la maggior difficoltà del compito dovuta alla presenza di più robot che devono coordinare i movimenti, il limite massimo per terminare ogni trial è stato fissato a 1 minuto anzichè 20 secondi. Una popolazione di 80 cromosomi, composti da 69 geni, è stata evoluta per 15000 generazioni, con probabilità di crossover impostata al 10% e quella di mutazione al 6%; il numero di individui d'elite è 3. Il problema è molto simile a quello descritto nell'esperimento di Quinn [30], pertanto la fitness function usata per valutare gli individui è analoga:

$$F = C_p \left(\sum_{t=1}^{T} \left[f\left(d_t, D_{t-1}\right) \left(1 - tanh(s_t/2)\right) \right] \right), \tag{5.2}$$

dove C_p è un componente che penalizza le collisioni tale che, se c è il numero medio[4] di collisioni tra robot, e c_{max} il massimo numero di collisioni consentito, allora $C_p = 1 - c/\left(2c_{max}\right)$. Nell'esperimento in questione c_{max} è fissato a 20. Al massimo C_p può dimezzare il punteggio ottenuto, in modo non penalizzare troppo controllori che magari collidono parecchie volte, ma sono in grado di eseguire il compito.
T è il numero di passi di simulazione eseguiti, siccome la durata di ognuno è di 0,1 secondi, il valore massimo è 600. $f\left(d_t, D_{t-1}\right)$ è un componente che premia il movimento del centroide rispetto alla sorgente luminosa. Se d_t è la distanza tra luce e centroide al tempo t e D_0 il suo valore per $t = 0$, allora:

$$f\left(d_t, D_{t-1}\right) = \begin{cases} \left(D_{t-1} - d_t\right) / \left(D_0 - 5\right) & \text{se } D_{t-1} > d_t \\ 0 & \text{altrimenti} \end{cases}, \tag{5.3}$$

[4]Usare la media del numero di collisioni velocizza il conteggio e ha lo stesso effetto che contare le collisioni separatamente per ciascun robot

se il trial prevede che il gruppo effettui fototassi; in questo caso D_{t-1} è il più piccolo valore mai raggiunto da d_t.
Se invece il gruppo deve allontanarsi dalla luce:

$$f(d_t, D_{t-1}) = \begin{cases} (d_t - D_{t-1})/D_f & \text{se } D_{t-1} < d_t,\ D_f = 0.5D_0 \\ 0 & \text{altrimenti} \end{cases}, \qquad (5.4)$$

con D_{t-1} che in questo caso è il valore più grande mai raggiunto da d_t.
Infine l'ultimo componente di F, s_t, serve a penalizzare la dispersione del gruppo di robot: se tutti i robot sono nel campo visivo dei sensori di prossimità di almeno un altro robot (5 cm col modello utilizzato), allora s_t vale 0 altrimenti vengono individuate le due linee più corte che connettono i robot ed s_t è la misura per la quale la più lunga delle due supera il raggio dei sensori. La costante 2 (nella formula 5.2) serve a regolare la pendenza della funzione *tanh*; valori più bassi penalizzano maggiormente la dispersione. Inizialmente si era usato un valore maggiore, ma i robot tendevano a separarsi troppo quindi si è deciso di pesare maggiormente s_t.
Dieci esperimenti evolutivi sono stati eseguiti con queste condizioni, la figura 5.6 mostra l'andamento della fitness della migliore evoluzione.

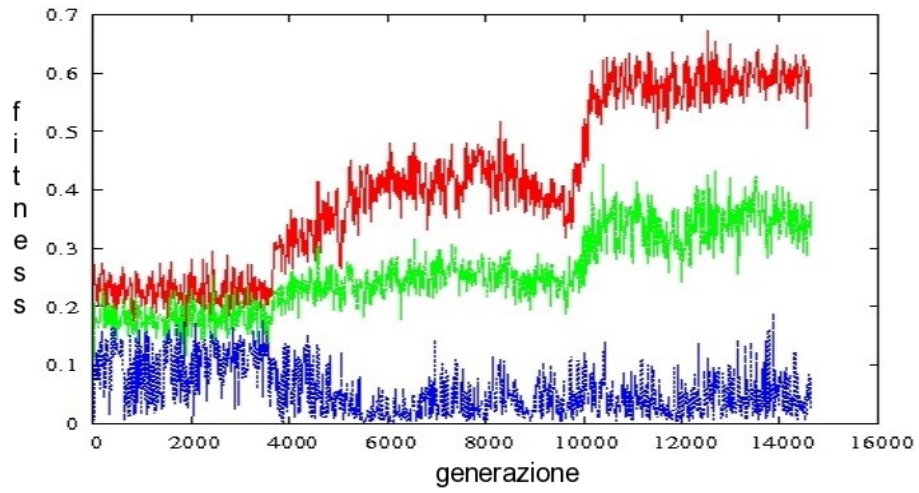


Figura 5.6: Andamento della fitness F, demonstrator equipaggiato con sensori luminosi di tipo lineare. Il grafico riporta i valori massimo, medio e minimo di ogni generazione.

A differenza del caso descritto precedentemente, i controllori sono stati valutati direttamente sul compito nel suo complesso, evitando la fase preliminare in cui si evolve la sola fototassi. Il grafico mostra tre momenti distinti nell'evoluzione del controllore. Analizzando il miglior individuo si può osservare

che inizialmente il controllore ha performance molto basse, dovute alla complessità del compito che richiede movimento di gruppo e attenzione a luce e suono. Attorno alla generazione 4000 le prestazioni iniziano a migliorare. Osservando il comportamento degli individui si nota che i robot stanno in gruppo e seguono la luce, ma non prestano attenzione al suono; infatti, la fitness si assesta attorno a 0,4 valore vicino a 0,45 che corrisponde al massimo ottenibile da un gruppo che segua sempre e comunque la luce, indipendentemente dal suono, e quindi svolga correttamente il compito in metà dei casi. Infine, attorno alla generazione 10000 si ha un aumento brusco della fitness, segnale che il controllore è in grado di apprendere la relazione tra luce e suono. L'analisi visiva conferma l'ipotesi. Tuttavia, il modo con cui i robot si muovono non è risultato soddisfacente come nell'esperimento descritto precedentemente: sia nell'avvicinarsi che nell'allontanarsi dalla luce i robot compiono delle rotazioni sul proprio asse. Inoltre, nei casi in cui la sorgente luminosa è più lontana, i robot faticano ad individuarla a causa del basso livello di attivazione dei sensori, che può essere coperto dal rumore.
Per far fronte al problema c'erano due soluzioni alternative: ritornare al modello a caratteristica lineare, oppure cercare di variare la procedura di sampling in modo da individuare condizioni che consentissero di avere un buon gradiente luminoso, che i robot potessero seguire con facilità. Delle due soluzioni è stata scelta la prima; il motivo è che non c'era la certezza che un comportamento di apprendimento sociale potesse essere evoluto, pertanto si è deciso di semplificare al massimo il problema per poi, eventualmente, ritornare a modelli più sofisticati se si fossero ottenuti dei risultati.
Si è deciso pertanto di eseguire una serie di esperimenti evolutivi in cui i robot erano equipaggiati con sensori di luminosità a caratteristica lineare. Come accennato in percedenza questi esperimenti hanno dato esiti positivi, ma affetti da alcuni problemi, pertanto sono stati poi rieseguiti con sensori di luminosità a caratteristica binaria e variando il regime evolutivo. La sezione seguente descrive gli esperimenti in questione.

5.4.3 Evoluzione del demonstrator con sensori di luminosità lineari e binari

Mentre nel caso di sensori di luminosità basati su lookup table la morfologia (orientamento e raggio) dei sensori di luminositá é definita implicitamente dai dati campionati dal robot e-puck, nel caso si intendano utilizzare altri modelli c'è la possibilità di scegliere posizione e raggio di tali sensori. Pertanto, allo scopo individuare la morfologia migliore per i sensori di luminosità, sono stati condotti degli esperimenti evolutivi in cui variano l'ampiezza del-

l'angolo visivo dei sensori (30°,45° e 60°) e la loro posizione sul robot (±30°, ±60°, ±90°). Dai risultati di questi esperimenti preliminari risulta che la combinazione che sembra migliore per poter seguire la luce è avere i sensori a ±90°, con un ampiezza di 60°.
A valle di questa analisi sono state condotte dieci evoluzioni, con i sensori di luminosità a caratteristica lineare configurati come descritto e con campo visivo di 1,5 m.
La figura 5.7 mostra l'andamento della fitness in una delle migliori evoluzioni.

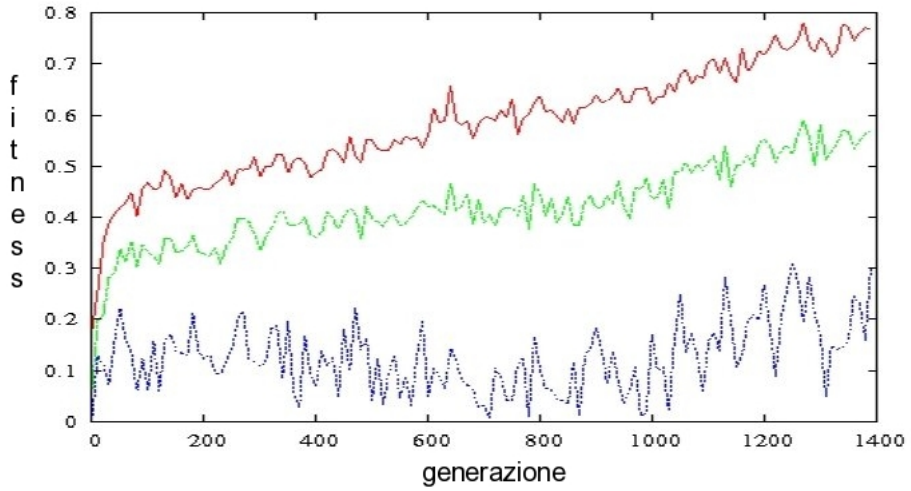


Figura 5.7: Andamento della fitness F nell'evoluzione del demonstrator, nel caso il robot sia equipaggiato con sensori luminosi a caratteristica lineare. Il grafico riporta i valori massimo, medio e minimo di ogni generazione.

Dal grafico si nota che i migliori controllori delle ultime generazioni hanno performance molto buone sebbene nessuno raggiunga il valore massimo 0,9. Questo è dovuto al fatto che i robot sono penalizzati quanto più si allontanano l'uno dall'altro e quanto più collidono. Per quanto bene svolgano il compito, ci sarà comunque una penalizzazione dovuta ad uno dei due aspetti.
Il miglior individuo trovato è stato utilizzato come demonstrator per alcuni degli esperimenti di social learning (ovvero in quelli dove i robot sono equipaggiati con sensori luminosi a caratteristica lineare). Anche se dai risultati di questi esperimenti è risultato che i robot sono in grado di svolgere il compito, é emerso un problema: i robot tendono a passare dal comportamento di fototassi a quello opposto anche senza aver percepito il suono (vedere sezione 5.5.2, in seguito, per maggiori dettagli). Il motivo di questo comportamento indesiderato è attribuibile al fatto che non c'è stata suffi-

ciente variabilitá in fase evolutiva: si ricorda che il segnale sonoro é percepito nei trial 4°, 5°, 6° o 7° nel gruppo di 10 (vedere tabella 5.1, pagina 48). La conseguenza é che strategie che portano la rete a modificare il comportamento, da fototassi ad antifototassi, dopo 4 o 5 trial anche senza la percezione del suono, permettono di ottenere buoni punteggi di fitness e pertanto si diffondono rapidamente nella popolazione, impedendo lo sviluppo di strategie ottime.
Per far fronte al problema sono stati condotti ulteriori esperimenti evolutivi utilizzando una distribuzione diversa dei trial di fototassi e di antifototassi (vedere tabella 5.2).

Tabella 5.2: Suddivisione dei trial, dopo l'aggiunta di gruppi di trial di sola fototassi e sola antifototassi. G_i indica l'i-esimo gruppo di 10 trials, T_i l'i-esimo trial nel gruppo, P un trial in cui il robot deve effettuare fototassi, S il trial in cui il suono viene percepito e A un trial in cui il robot deve fare antifototassi.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
G_1-G_2	P	P	P	**S**	A	A	A	A	A	A
G_3-G_4	**S**	A	A	A	A	A	A	A	A	A
G_5-G_6	P	P	P	P	P	P	P	P	P	**S**
G_7-G_8	P	P	P	P	P	P	**S**	A	A	A

La differenza risiede nel fatto che sono stati aggiunti gruppi di trial in cui il suono viene prodotto da subito (G_3 e G_4) oppure alla fine (G_5 e G_6). Ciò ha portato a notevoli miglioramenti delle prestazioni nel compito di apprendimento individuale (vedere sezione 5.5.3), in quanto è stato eliminato il fenomeno per cui i robot modificano autonomamente il comportamento, senza aver percepito il suono. I risultati delle analisi nella tabella 5.7 (pagina 66) confermano l'affermazione.
In questi esperimenti il demonstrator é equipaggiato con sensori di luminosità di tipo binario (l'output del sensore è 1 se la luce è nel campo visivo, 0 altrimenti; vedere sezione 4.3, pagina 35). Il motivo di questa variazione sta nel fatto che tale modello di sensore è facilmente portabile sull'e-puck usando un meccanismo a soglia.
La figura 5.8 mostra l'andamento della fitness nelle 5000 generazioni create dall'algoritmo genetico.
Dall'analisi del grafico si può notare la presenza di un brusco cambiamento nel valore della fitness attorno alla generazione 3000, che passa da valori prossimi a 0,5 a valori ben più alti. Questo denota la comparsa di individui in grado di associare il suono al movimento da tenere nei confronti della luce. I migliori controllori sono in grado di svolgere il compito in modo soddisfacente, superando i valori di fitness ottenuti nel caso in cui i robot erano equipaggiati con sensori luminosi a caratteristica lineare. Anche in questo

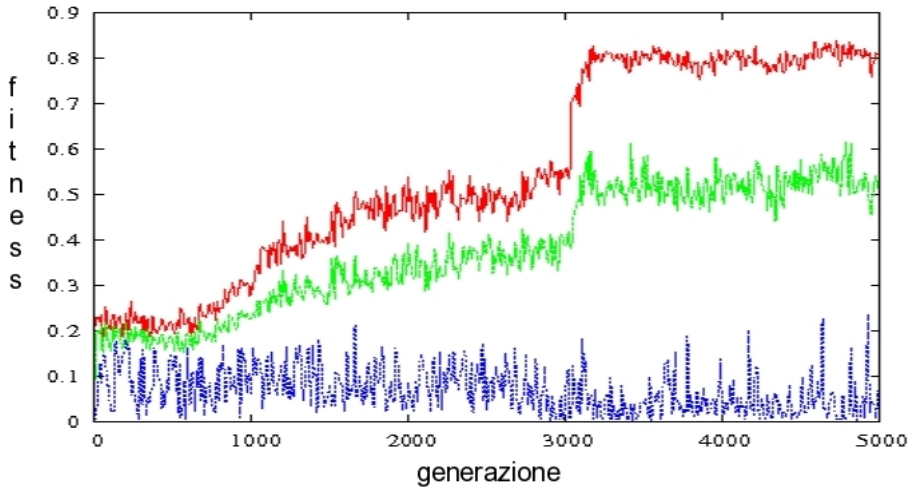


Figura 5.8: Andamento della fitness F, demonstrator equipaggiato con sensori luminosi a soglia. Il grafico riporta i valori massimo, medio e minimo di ogni generazione.

caso il miglior individuo ottenuto è stato utilizzato come demonstrator negli esperimenti di social learning.
La sezione seguente descrive gli esperimenti evolutivi condotti per evolvere la rete che controlla il learner, in grado di apprendere sia dal suono che dall'interazione col demonstrator.

5.5 Evoluzione del learner

Questa sezione descrive gli esperimenti evolutivi condotti con lo scopo di evolvere il controllore del learner, ovvero una rete ricorrente in grado di svolgere entrambi i compiti di apprendimento. Come nel caso dell'evoluzione del demonstrator é stato condotto un primo esperimento evolutivo, di carattere esplorativo, il cui scopo era cioè di individuare un buon setup sperimentale. La sezione seguente descrive questo primo esperimento.

5.5.1 Primo esperimento evolutivo

In questo esperimento una popolazione di 80 individui è stata generata casualmente, il tasso di crossover settato al 5% e la probabilità di mutazione all'8%; il numero di individui d'elite 3.
L'esperimento prevede due fasi distinte. Inizialmente sono presenti entrambi i robot, learner e demonstrator, ad una distanza di 1 cm l'uno dall'altro ed

in modo che resti libero un angolo di 90° tra demonstrator e luce[5]. Lo stato dei controllori dei due robot é diverso: i livelli di attivazione dei neuroni del demonstrator sono caricati da un file, quelli dei neuroni del learner sono impostati a zero (rete resettata). Cosí facendo é possibile fare in modo che il demonstrator si muova nel modo desiderato: due file contengono livelli di attivazione che portano la rete a compiere, rispettivamente, fototassi e antifototassi, caricando uno dei due file si ottiene il comportamento voluto[6]. La rete del learner è invece resettata perchè sia in condizioni "neutre", che consentano l'apprendimento.
Nella prima fase dell'esperimento entrambi i robot sono liberi di muoversi. Il demonstrator, a seconda dello stato caricato dal file, si avvicinerà o allontanerà dalla luce; compito del learner è seguirne i movimenti, usando i dati dei sensori di prossimità. Quando il demonstrator si è avvicinato a meno di 5 cm dalla sorgente luminosa o a superato una distanza di una volta e mezza quella che inizialmente lo separava dalla luce, la prima fase dell'esperimento termina ed inizia la seconda.
Nella seconda fase il demonstrator viene rimosso e la luce riposizionata ad una distanza compresa tra i 55 e 65 cm dal learner. A questo punto il learner è solo e deve ripetere quanto osservato in precedenza dal demonstrator: avvicinarsi alla luce se il demonstrator aveva effettuatato fototassi o allontanarsi in caso contrario. Dato che l'obiettivo è ottenere una rete che sia in grado di apprendere sia individualmente che socialmente, la valutazione di ogni individuo viene condotta su entrambi i compiti.
Gli individui sono stati valutati per 92 volte ciascuno: le prime 32 valutazioni riguardano l'apprendimento individuale, la funzione di fitness usata è la stessa descritta in precedenza (sezione 5.1, pag. 49). Stavolta i trial sono organizzati in gruppi di 8, col suono prodotto al 4° o al 5° trial del gruppo[7]. La tabella 5.3 mostra l'organizzazione di questi 32 trial.

Tabella 5.3: Suddivisione dei trial di apprendimento individuale. G_i indica l'i-esimo gruppo di 8 trials, T_i l'i-esimo trial nel gruppo, P un trial in cui il robot deve effettuare fototassi, S il trial in cui il suono viene percepito e A un trial in cui il robot deve fare antifototassi.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
G_1-G_2	P	P	P	**S**	A	A	A	A
G_3-G_4	P	P	P	P	**S**	A	A	A

[5]Così facendo il learner non ostacola i movimenti del demonstrator

[6]I dati nei due file provengono da trial di apprendimento individuale, prima e dopo che il demonstrator abbia percepito il suono

[7]Si ricorda che questi esperimenti sono stati condotti prima di introdurre il regime riassunto nella tabella 5.2

Nelle rimanenti 60 prove, divise equamente in fototassi ed antifototassi, viene condotto l'esperimento di apprendimento sociale descritto precedentemente in questa sezione.
La fitness di ogni cromosoma si compone di due parti, la prima (F_{sl1}) premia gli individui che più restano vicini al demonstrator durante la fase in cui esso è presente ed è calcolata come:

$$F_{sl1} = C_p(0.9 - d_m/100 + 0.1), \tag{5.5}$$

dove il termine C_p è lo stesso usato nell'equazione 5.2 per penalizzare le collisioni; anche in questo caso il numero massimo di collisioni consentite, prima di fermare il trial, è 20. d_m è la massima distanza (in cm) tra i due robot durante il trial, essa viene impostata a 0 se non supera il raggio dei sensori ad infrarossi (5 cm col modello utilizzato). Il valore 100 è il massimo valore tollerato per d_m, sopra il quale il trial viene fermato. Infine, la costante 0,1 è un bonus che viene dato solo nei seguenti casi:

- Il learner termina un trial in cui il demonstrator ha effettuato fototassi a meno di 5 cm dalla luce;
- Il learner termina un trial di antifototassi a più di una volta e mezza la distanza iniziale dalla sorgente luminosa;

il valore finale viene poi normalizzato tra 0 ed 1 in quanto potrebbe assumere valori negativi.
Quando il learner viene lasciato solo, la fitness è calcolata in modo diverso, in quanto vanno valutate le capacità di apprendimento sociale:

$$F_{sl2} = \begin{cases} \frac{d_{i5} - d_f}{d_{i5}} & \text{nei trial di fototassi} \\ \frac{d_f - d_i}{0.5d - i} & \text{nei trial di antifototassi} \end{cases}, \tag{5.6}$$

dove d_i è la distanza iniziale, d_{i5} la distanza iniziale ridotta di 5 cm, d_f la distanza finale tra robot e luce. Il punteggio finale che un individuo ottiene è dato dalla media sui 92 trial.
Nessuno dei dieci esperimenti evolutivi condotti in queste condizioni ha portato a risultati apprezzabili, dunque si è deciso di cambiare il modo di operare.
La sezione seguente descrive gli esperimenti che hanno portato all'evoluzione di reti in grado di controllare il learner nei due compiti di apprendimento, sia nel caso sia equipaggiato di sensori di luminosità a caratteristica lineare sia che abbia sensori a caratteristica binaria.

5.5.2 Evoluzione del learner con sensori di luminosità lineari

Siccome il precedente tentativo di evolvere un buon learner a partire da una popolazione generata casualmente non ha portato nessun risultato, si è deciso di affrontare il problema in maniera diversa. La strategia adottata deriva da un confronto tra la natura e l'approccio usato nell'esperimento descritto. In natura, le specie che mostrano comportamenti di apprendimento sociale hanno già le capacità per svolgere il compito che stanno imitando. Questo vuol dire che la complessità da parte di chi deve apprendere sta nel capire, durante l'imitazione, quando convenga effettuare un compito per svolgere il quale si hanno già le capacità cerebrali e motorie. Con l'approccio descritto, non solo l'evoluzione deve sviluppare individui che imitano (seguendo) il demonstrator, ma anche i meccanismi che fanno si che il robot presti attenzione alla luce e si muova di conseguenza. A valle di questa analisi si è deciso di inizializzare la prima generazione non in modo casuale, ma partendo dal demonstrator modificandone i geni in modo lieve. In questo modo la popolazione di partenza ingloba già il fatto di direzionare il movimento sulla base della sorgente luminosa, mentre il compito dell'algoritmo evolutivo è sviluppare i meccanismi per seguire il demonstrator ed interpretarne i movimenti, mantenendo anche la capacità di apprendere individualmente dal suono.
Una popolazione di 80 individui è stata creata inizializzando ogni individuo a partire dal demonstrator ed aggiungendo ad ognuno dei geni un offset generato casualmente nell'intervallo [-0.1,0.1]. Poichè il demonstrator usato è equipaggiato con sensori di luminosità a caratteristica lineare, anche il learner è tale, poichè generato per modifiche del demonstrator. Il numero di individui d'elite è stato impostato a 3, la probabilità di crossover al 5% e quella di mutazione all'8%. Il regime di valutazione di ogni cromosoma è lo stesso del caso descritto in precedenza, ovvero 92 trial di cui 32 di apprendimento individuale e 60 di apprendimento sociale. In una prima serie di 10 evoluzioni nessuno dei controllori riusciva a svolgere il compito. Osservando il comportamento si notava che i robot erano in grado di apprendere individualmente tramite il suono, ma non attraverso l'imitazione del demonstrator. Il problema è stato risolto pesando maggiormente il punteggio ottenuto nei trial di apprendimento sociale: i valori ottenuti con le formule 5.5 e 5.6 sono stati moltiplicati per 6[8]; così facendo, la pressione selettiva nei confronti di controllori capaci di svolgere anche solo in maniera parziale il compito di apprendimento sociale era maggiore. Anche con questo nuovo setup il

[8]Costante individuata attraverso prove sperimentali

numero di evoluzioni condotte è 10, la figura 5.9 mostra l'andamento della fitness del miglior individuo di ognuna.

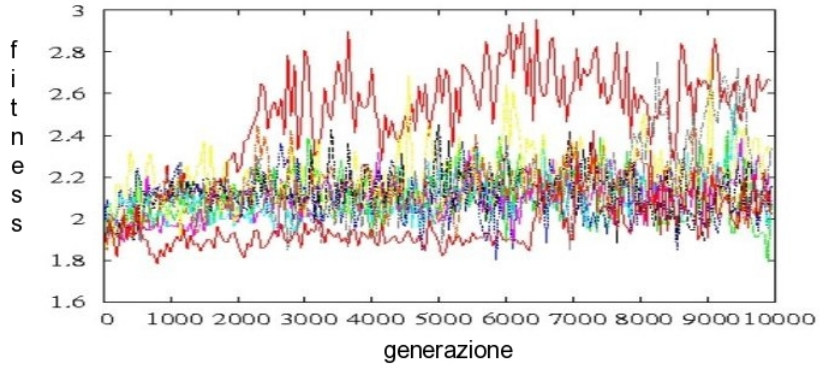


Figura 5.9: Andamento della fitness dei migliori individui nelle 10 evoluzioni. I robot sono equipaggiati con sensori di luminosità a caratteristica lineare.

Dal grafico si nota che una delle evoluzioni ha prodotto dei controllori con performance superiori rispetto a quelle delle altre 9. Il valore della fitness nelle prime generazioni si assesta in un intorno di 2; questo si può spiegare considerando che il punteggio massimo ottenibile nei 92 trial è circa 4,2. Dato che la popolazione iniziale è creata a partire dal demonstrator gli individui sono in grado di svolgere il compito di apprendimento individuale, pertanto è presumibile che all'inizio effettueranno di preferenza fototassi in attesa del suono, anche quando affiancati dal demonstrator. Ciò vuol dire che dei 60 trial di apprendimento sociale risolveranno in modo corretto metà dei casi: nel caso di apprendimento sociale il suono non viene prodotto, pertanto se i robot si muovo in attesa del suono effettueranno sempre fototassi, risolvendo correttamente i casi in cui anche il demonstrator effettua fototassi e fallendo nell'altra metà in cui il demonstrator effettua antifototassi. Ne consegue che il punteggio che otterranno sarà circa metà del massimo possibile, ovvero 2,1. Per quanto riguarda i picchi nel grafico essi raggiungono valori in un intorno di 3 e corrispondono a controllori in grado di svolgere il task in modo corretto, nei casi di apprendimento individuale e sociale. I punteggi che si osservano sono abbastanza lontani dai valori limite teorici in quanto le componenti che penalizzano la distanza tra i robot e le collisioni influenzano la valutazione: infatti, risulta difficile per il learner sia non collidere sia stare abbastanza vicino al demonstrator da non uscire mai dal raggio dei sensori ad infrarossi.

Valutazione del comportamento ottenuto

Il miglior individuo ottenuto è stato testato nei due compiti, per valutarne le prestazioni. La procedura di valutazione dell'apprendimento individuale prevede che il robot affronti dei trial raggruppati in insiemi di 8 (ogni 8 trial il controllore viene resettato e la memoria cancellata). Ognuno di questi insiemi di trial si differenzia dagli altri per il trial in cui viene prodotto il suono: nell'insieme S_1 il suono è percepito nel primo trial del gruppo, in S_2 nel secondo e così via fino a S_8 col suono all'ottavo, e ultimo, trial dell'insieme. Obiettivo della suddivisione è verificare se le capacità di apprendimento dipendono da quando il suono viene prodotto. Ogni controllore è stato valutato 10000 volte per ognuno dei set S_i, per un totale di 640000 trial; per ogni set è stata registrata la percentuale di successo e quella d'errore. Notare che gli errori possono essere di due tipi:

1. Il robot effettua antifototassi, ma il trial prevede che si avvicini alla luce. Questa tipologia di errore è etichettata con E_1.

2. Il robot approccia la sorgente luminosa, ma il trial corrente prevede che si allontani. Questa tipologia di errore è etichettata con E_2.

La tabella 5.4 mostra le percentuali di successo S e quelle di errore E_1 ed E_2, per ognuno dei gruppi S_i. In tutti i casi il trial in cui il suono viene percepito non compare nelle percentuali.

Tabella 5.4: Risultati della valutazione dell'apprendimento individuale. S è la percentuale di successo, E_1 la percentuale degli errori di tipo 1 ed E_2 quella degli errori di tipo 2. S_i indica il tipo di set. Il robot è equipaggiato con sensori di luminosità a caratteristica lineare; il controllore è evoluto secondo il regime descritto nella tabella 5.1, pagina 48. I risultati si riferiscono al miglior controllore mai evoluto in querse condizioni.

S_1			S_2			S_3			S_4		
S	E_1	E_2	S	E_1	E_2	S	E_1	E_2	S	E_1	E_2
99.5	0.00	0.50	95.6	2.80	1.60	92.7	6.20	1.10	88.8	10.5	0.70
S_5			S_6			S_7			S_8		
S	E_1	E_2	S	E_1	E_2	S	E_1	E_2	S	E_1	E_2
84.3	15.3	0.40	77.6	22.2	0.20	69.8	30.1	0.10	60.7	39.3	0.00

Si è scelto di separare i dati di ogni set in modo da valutare la "stabilità" del comportamento: chiaramente sono diverse le situazioni in cui il robot percepisce il suono all'inizio del set rispetto a quelle in cui lo percepisce nei trial finali o centrali.
Dai dati raccolti si può concludere che il controllore è in grado di apprendere la relazione tra suono e movimento rispetto alla luce in maniera sod-

disfacente, come mostrato dalle buone percentuali di successo S. Tuttavia si può notare che c'è una tendenza innata a cambiare comportamento, da fototassi ad antifototassi, in maniera autonoma. Questa osservazione è supportata dal fatto che le percentuali di errore di tipo E_1 (il comportamento atteso è approcciare la sorgente luminosa, ma il robot si allontana) sono tanto più alte tanto più il suono viene prodotto tardi nel set di prove. Ciò indica che il robot modifica il proprio comportamento in modo autonomo, senza aver percepito il segnale sonoro. Questo risultato può essere spiegato considerando che durante l'evoluzione, nei trial di apprendimento individuale, la rete ha ricevuto l'input dal microfono solo nel 4° o 5° trial di ciascun gruppo (con gruppi di 8, vedere tabella 5.3, pagina 57). L'evoluzione ha quindi trovato una soluzione in cui la rete in qualche modo è in grado di percepire lo scorrere del tempo e modificare il comportamento attorno al 4°/5° trial; un differente regime evolutivo (tabella 5.2, pagina 55) consente di eliminare questo fenomeno (vedere i risultati nella sezione seguente).
Ciononostante si può notare che il controllore è in grado di apprendere dal suono, infatti le percentuali di errore E_2 (il comportamento atteso è allontanarsi dalla sorgente luminosa, ma il robot si avvicina) sono molto basse anche quando il segnale sonoro è percepito nei primi trial del set, fino a raggiungere valori quasi nulli quando il suono è prodotto molto tardi nel set di prove [9]. Per quanto riguarda la valutazione delle capacità di apprendimento sociale, i trial sono organizzati in gruppi di 9, secondo la seguente procedura:

1. Inizialmente entrambi i robot (learner e demonstrator) sono presenti nell'arena, il demonstrator mostra il comportamento da tenere (fototassi o antifototassi);

2. In un secondo momento il demonstrator viene rimosso ed il learner deve ripetere il comportamento osservato. Ogni volta che la distanza dalla luce è minore di 5 cm o maggiore di 1,5 volte la distanza iniziale il learner viene riposizionato al centro dell'arena e la luce cambia posizione;

Il passo 2 viene ripetuto per 8 volte, dopodichè gli stati di attivazione dei neuroni vengono impostati a zero e si ricomincia dal passo 1. Il demonstrator al passo 1 può compiere fototassi o antifototassi, a seconda dei livelli di attivazione che vengono caricati nei neuroni.
Il ciclo è stato iterato per 10000 volte per entrambi i comportamenti, per un totale di 80000 prove in cui ci si attende un comportamento di fototassi ed

[9] Il set S_8 non va considerato in quanto non è possibile che il robot commetta errori di tipo E_2

80000 in cui il comportamento atteso è l'opposto.
La tabella 5.5 mostra i risultati delle valutazioni.

Tabella 5.5: Risultati delle valutazioni. La tabella mostra le percentuali di successo S e di errore E nei trial che richiedono fototassi P_i e antifototassi A_i.

	P_1	A_1	P_2	A_2	P_3	A_3	P_4	A_4
S	87.0	83.2	82.7	84.4	75.7	86.1	65.8	88.0
E	13.0	16.8	17.3	15.6	24.3	13.9	34.2	12.0
	P_5	A_5	P_6	A_6	P_7	A_7	P_8	A_8
S	55.4	90.2	44.8	92.3	35.4	94.1	27.2	95.5
E	44.6	9.80	55.2	7.70	64.6	5.90	72.8	4.50

Le righe S ed E indicano rispettivamente le percentuali di successo e di errore. Le colonne etichettate con P_i indicano trial in cui ci si attende che il learner effettui fototassi, le colonne A_i quelli in cui il comportamento atteso è di antifototassi. L'indice i ($i \in [1, 8]$) indica il numero del trial nel gruppo di 8. Pertanto P_1 indica una prova che segue immediatamente la rimozione del demonstrator e in cui ci si attende un comportamento di fototoassi, P_2 una prova che segue un trial di tipo P_1 e così via fino a P_8; in modo analogo vanno interpretate le colonne etichettate con A_i (unica differenza: il comportamento atteso è anitifototassi). Questa suddivisione è stata adottata per verificare la stabilità del comportamento appreso.
Dai dati raccolti, si può concludere che il controllore è in grado di svolgere correttamente il compito di apprendimento individuale: le colonne P_1 ed A_1 (ovvero nei trial immediatamente successivi la rimozione del demonstrator) mostrano percentuali di successo superiori all'80%. Il fatto che nel caso di fototassi le prestazioni siano migliori è dovuto alla struttura del task. Infatti il robot è portato per "natura" a essere attratto dalla luce: se il demonstrator è assente il learner si dirige verso la sorgente luminosa in attesa del suono. Alla luce di questa considerazione l'83% di successo nel caso di antifototassi, subito dopo la rimozione del demonstrator (colonna A_1), rappresenta un risultato molto positivo perché il robot va "contro natura", dopo aver seguito il demonstrator, ciò denota senza alcun dubbio la capacità di apprendere socialmente.
Tuttavia, è ancora presente il fenomeno descritto in precedenza, ovvero la tendenza a modificare il comportamento da fototassi ad antifototassi. Ciò è messo in evidenza dalle pessime prestazioni nei trial di tipo P_7 e P_8 e dalle prestazioni ottime nei corrispettivi trial A_7 ed A_8: con lo scorrere del tempo la tendenza a eseguire antifototassi cancella quanto appreso dal demonstrator (o dualmente lo rinforza). Questo comportamento è dovuto,

come spiegato, al tipo di regime utilizzato per valutare la parte di apprendimento individuale durante la fase evolutiva.
La sezione seguente descrive gli esperimenti evolutivi condotti per ottenere un learner equipaggiato con sensori di luminosità a caratteristica binaria.

5.5.3 Evoluzione del learner con sensori di luminosità binari

Visti i problemi individuati dopo l'analisi dei risultati, si è deciso di modificare il regime evolutivo per far fronte alla tendenza della rete a passare al comportamento di antifototassi in modo autonomo. Anche i sensori con cui sono equipaggiati i robot sono diversi rispetto al caso precedente: qui i sensori di luminosità sono a caratteristica binaria anzichè lineare (vedere sezione 4.3, pagina 35), decisione presa per poter portare eventuali risultati sul robot e-puck in cui i sensori non hanno caratteristiche lineari.
Anche in questo caso la popolazione iniziale è stata generata a partire dal cromosoma del demonstrator. Una prima differenza a livello di regime evolutivo è nell'organizzazione dei primi 32 trial (in cui viene valutata la capacità di apprendimento individuale): come per il caso dell'evoluzione del demonstrator sono stati introdotti gruppi di trial di sola fototassi (G_3) e sola antifotassi (G_2). La tabella 5.6 mostra a suddivisione adottata.

Tabella 5.6: Suddivisione dei trial di apprendimento individuale. G_i indica l'i-esimo gruppo di 8 trials, T_i l'i-esimo trial nel gruppo, P un trial in cui il robot deve effettuare fototassi, S il trial in cui il suono viene percepito e A un trial in cui il robot deve fare antifototassi.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
G_1	P	P	P	**S**	A	A	A	A
G_2	**S**	A	A	A	A	A	A	A
G_3	P	P	P	P	P	P	P	**S**
G_4	P	P	P	P	**S**	A	A	A

Anche a livello di fitness function si sono introdotte delle modifiche. La differenza risiede nel fatto che i 32 trial di apprendimento individuale non assegnano fitness: ogni individuo viene valutato su questi trial, al termine dei quali non riceve alcun punteggio, ma viene tenuta traccia del numero di volte in cui il compito è svolto correttamente. Una prova è contata come successo quando il robot deve effettuare fototassi e termina a meno di 5 cm dalla luce, oppure quando finisce ad una distanza maggiore di una volta e mezza quella iniziale e la prova richiede antifotoassi.
Solo gli individui che hanno successo in almeno 25 prove su 32 (i trial col suono non vengono contati) possono procedere con le restanti prove di apprendimento sociale e quindi ricevere un punteggio. In questa maniera si

mantiene la capacità di apprendere individualmente mentre viene evoluta quella di apprendere socialmente.
La figura 5.10 mostra l'andamento della fitness del miglior individuo di ogni evoluzione.

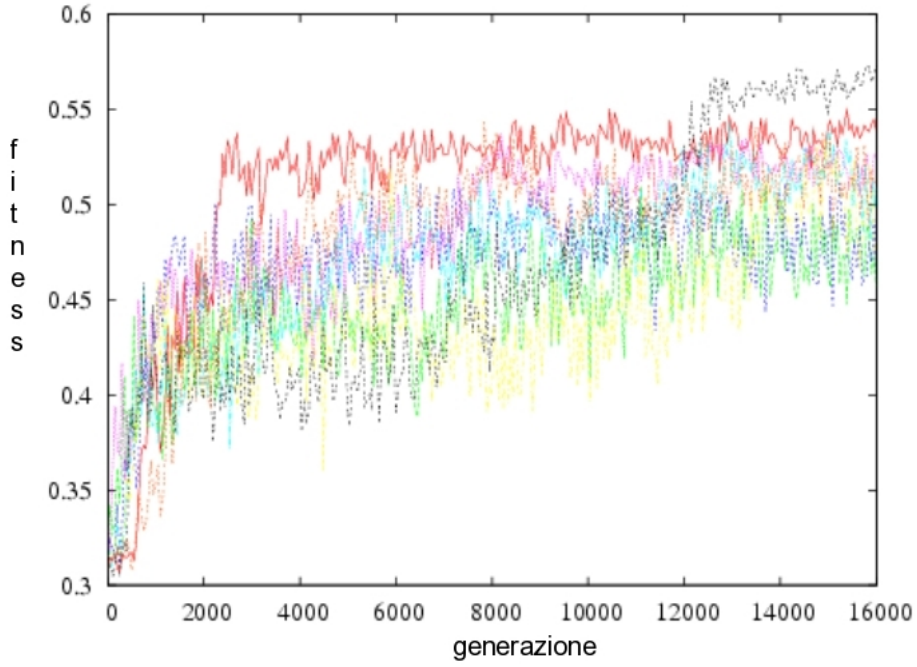


Figura 5.10: Andamento della fitness del miglior individuo. I robot sono equipaggiati con sensori di luminosità ad output binario

Dal grafico si nota come tutte le evoluzioni producano controllori in grado di svolgere i compiti di apprendimento con buone prestazioni. I punteggi dei migliori controllori si assestano, nelle ultime generazioni, intorno a 0.5, un valore molto buono considerando un massimo di circa 0.65 [10]. In questo caso non si è rivelato necessario pesare in modo maggiore alcuna delle componenti delle fitness function 5.5 e 5.6.

Valutazione del comportamento ottenuto

Anche in questo caso sono stati condotti degli esperimenti per valutare le prestazioni dei controllori ottenuti. I test seguono la stessa procedura descritta nella sezione 5.5.2 (pag. 61). Le evoluzioni condotte in queste condizioni hanno prodotto dei risultati migliori, pertanto sono stati valutati i

[10] I 60 trial di apprendimento sociale danno punteggio massimo 1, i 32 di apprendimento individuale non danno punteggio

migliori controllori di ogni evoluzione (8 in totale). La tabella 5.7 mostra i risultati ottenuti, C_i indica il miglior individuo dell'evoluzione i-esima, C_T i risultati complessivi dei controllori testati. Poichè questi controllori si sono rivelati altamente performanti, anzichè presentare la percentuale di successo e di errore di tipo 1 e 2, sono indicati il numero complessivo di errori dei due tipi sul totale di 70000 trial (colonne NE_1 ed NE_2). Le percentuali di errore sono infatti molto basse (prossime a 0) e non consentono di apprezzare le differenze tra i vari controllori.

Tabella 5.7: Risultati della valutazione dell'apprendimento individuale. NE_1 indica il numero di errori di tipo 1 ed NE_2 quello di tipo 2. S_i indica il tipo di set. Il robot è equipaggiato con sensori di luminosità a caratteristica binaria; il controllore è evoluto secondo il regime descritto nella tabella 5.2, pagina 55

	S_1		S_2		S_3		S_4	
	NE_1	NE_2	NE_1	NE_2	NE_1	NE_2	NE_1	NE_2
C_0	0	0	1	0	8	5	20	4
C_1	0	0	0	0	0	0	0	0
C_2	0	0	0	0	0	0	0	0
C_3	0	0	11	0	47	0	90	0
C_4	0	91	0	0	0	0	0	0
C_5	0	80	12	38	16	26	23	10
C_6	0	7	0	18	5	25	13	32
C_7	0	0	36	0	30	0	87	0
C_T	0	178	60	56	106	56	233	46
	S_5		S_6		S_7		S_8	
	NE_1	NE_2	NE_1	NE_2	NE_1	NE_2	NE_1	NE_2
C_0	18	12	46	2	69	1	91	0
C_1	0	0	0	0	0	0	0	0
C_2	0	0	0	0	0	0	0	0
C_3	150	0	245	0	305	0	453	0
C_4	0	0	0	0	0	0	0	0
C_5	28	12	24	12	38	3	72	0
C_6	13	9	22	4	32	5	34	0
C_7	144	0	155	0	150	0	126	0
C_T	353	33	492	18	594	9	776	0

Come si può notare c'è un miglioramento netto rispetto ai risultati riassunti nella tabella 5.4: raramente il controllore passa al comportamento di antifototassi senza prima aver percepito il segnale sonoro. Ciò conferma l'ipotesi che il problema delineato nella sezione precedente deriva dal regime evolutivo utilizzato e non da limiti intrinseci dovuti alla rete neurale. Inoltre a differenza del caso precedente, tutte le evoluzioni hanno portato a controllori in grado di svolgere il compito in maniera quasi perfetta, il numero totale di errori nella riga C_T è infatti molto basso per ogni gruppo S_i. Da notare che i migliori controllori (C_1 e C_2) non hanno sbagliato mai un singolo trial sul totale dei 70000 a disposizione. Anche la procedura di valutazione

delle capacità di apprendimento sociale è stata ripetuta, secondo la modalità descritta nella sezione 5.5.2. Le tabelle 5.8 e 5.9 riassumono i dati raccolti.

Tabella 5.8: Risultati delle valutazioni, i robot sono equipaggiati con sensori di luminosità a caratteristica binaria. La tabella mostra le percentuali di successo S e di errore E (percentuale di trial terminati lontano dalla luce) per ogni cromosoma C_i nei trial che richiedono fototassi.

		P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
C_0	S	96.68	96.66	96.61	96.58	96.55	96.52	96.45	96.4
	E	3.32	3.24	3.39	3.42	3.45	3.48	3.55	3.6
C_1	S	89.42	89.42	89.41	89.4	89.4	89.4	89.39	89.37
	E	10.58	10.58	10.59	10.6	10.6	10.6	10.61	10.63
C_2	S	80.98	80.96	80.92	79.98	79.95	79.89	78.83	78.75
	E	19.02	19.04	19.08	20.02	20.05	20.11	20.17	20.25
C_3	S	97.22	97.01	96.84	96.63	96.42	96.18	95.96	95.74
	E	2.78	2.99	3.16	3.37	3.58	3.82	4.04	4.26
C_4	S	95.5	95.39	95.26	94.98	94.83	94.64	94.55	94.17
	E	4.5	4.61	4.74	5.02	5.17	5.36	5.56	5.83
C_5	S	89.45	89.45	89.45	89.45	89.44	89.43	89.42	89.41
	E	10.55	10.55	10.55	10.55	10.56	10.57	10.58	10.59
C_6	S	99.24	99.2	99.13	99.21	99.18	99.16	99.16	99.15
	E	0.76	0.8	0.87	0.79	0.82	0.84	0.84	0.85
C_7	S	95.51	95.5	95.48	95.44	95.38	95.33	95.27	95.19
	E	4.49	4.5	4.52	4.56	4.62	4.67	4.73	4.81

Tabella 5.9: Risultati delle valutazioni, i robot sono equipaggiati con sensori di luminosità a caratteristica binaria. La tabella mostra le percentuali di successo S e di errore E (percentuale di trial terminati vicino alla luce) per ogni cromosoma C_i nei trial che richiedono antifototassi.

		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
C_0	S	97.13	97.14	97.15	97.15	97.15	97.16	97.16	97.16
	E	2.87	2.86	2.85	2.85	2.85	2.84	2.84	2.84
C_1	S	78.16	78.16	78.18	78.19	78.2	78.21	78.23	78.26
	E	21.84	21.84	21.82	21.81	21.8	21.19	21.17	21.14
C_2	S	91.67	91.7	91.78	91.88	92.01	92.12	92.25	92.39
	E	8.83	8.3	8.22	8.12	7.99	7.88	7.75	7.61
C_3	S	97.99	98.01	98.02	98.02	98.03	98.04	98.06	98.07
	E	2.01	1.99	1.98	1.98	1.97	1.96	1.94	1.93
C_4	S	93.1	93.13	93.15	93.19	93.21	93.25	93.31	93.39
	E	6.9	6.87	6.85	6.81	6.79	6.75	6.69	6.61
C_5	S	98.69	98.68	98.68	98.66	98.65	98.65	98.65	98.63
	E	1.31	1.32	1.32	1.34	1.35	1.35	1.35	1.37
C_6	S	98.05	98.05	98.06	98.06	98.07	98.08	98.09	98.1
	E	1.95	1.95	1.94	1.94	1.93	1.92	1.91	1.9
C_7	S	96.73	96.73	96.74	96.75	96.77	96.8	96.83	96.88
	E	3.27	3.27	3.26	3.25	3.23	3.2	3.17	3.22

La tabella mostra che i migliori individui di ogni esperimento evolutivo hanno buone prestazioni nel compito di apprendimento sociale. Curiosamente i peggiori controllori sono quelli che nel caso di apprendimento individuale

non hanno sbagliato un singolo trial di valutazione (C_1 e C_2).
Le migliori prestazioni sono dovute, come nel caso individuale, al fatto che non c'è più la tendenza a passare ad un comportamento di antifototassi autonomamente.

Capitolo 6

Risultati e conclusioni

Negli ultimi anni sempre più lavori hanno riguardato lo studio di aspetti legati all'apprendimento sociale da parte di robot o agenti autonomi in generale. Questo interesse è motivato dalla volontà di implementare agenti in grado di condividere conoscenza, apprendendo gli uni dagli altri attraverso l'interazione. Nei casi in cui l'informazione necessaria ad apprendere in maniera individuale sia difficilmente accessibile, oppure dove i costi legati all'apprendimento sono elevati, il fatto di avere elementi in grado di scambiarsi conoscenza rappresenta un vantaggio.
Molti lavori hanno portato a soluzioni più o meno complesse per affrontare il problema, con applicazioni pratiche quali la programmazione di manipolatori attraverso la dimostrazione o l'implementazione di comportamenti d'imitazione in robot mobili autonomi. In letteratura, tuttavia, non esistono esempi in cui l'approccio evolutivo viene usato per sviluppare controllori neurali in grado di apprendere in ambiente sociale. Questa tesi dimostra che tale metodologia può essere impiegata con successo per ottenere questi tipi di comportamento e che le reti ricorrenti sono sufficientemente flessibili da poter combinare entrambe le tipologie di apprendimento, individuale e sociale, in un unico controllore.
Questo lavoro conferma inoltre i risultati già ottenuti da Di Paolo in [13], da Tuci e Quinn in [33, 34] e Phattanasri, Chiel e Beer in [29]; ovvero che le reti ricorrenti possono essere impiegate con successo in compiti che richiedono coordinazione senso-motoria unita a capacità di apprendimento. I dati riassunti nelle tabelle 5.4 e 5.7 (capitolo precedente) supportano questa affermazione.
Al momento della stesura della tesi alcuni dei controllori evoluti sono stati trasferiti sui robot e-puck per verificare che quanto avviene in simulazione

funzioni anche in ambiente reale. Le prime impressioni sono positive, anche se si ha un calo di prestazioni dovuto ai casi in cui un robot fa ombra all'altro e quindi sporca le letture dei sensori. Questa situazione non è modellizzata nel simulatore e pertanto è comprensibile che il controllore si trovi in difficoltà. Nel complesso, il comportamento ottenuto è comunque abbastanza fedele a quello osservato in ambiente simulato, confermando l'applicabilità dell'approccio evolutivo per implementare controllori da utilizzare su robot reali.
Una possibile estensione al lavoro presentato potrebbe essere lo studio dell'impiego di reti plastiche per affrontare il medesimo task, valutando se esse siano in grado o meno di risolvere il problema e con quali prestazioni. Data la semplicità del compito in esame potrebbe essere opportuno definire una serie di task via via più complessi, in modo da stabilire se esistano differenze apprezzabili tra le due tipologie di controllori.
Un'altra estensione riguarda il passaggio ad un gruppo di robot omogeneo: nel caso qui presentato i due robot, learner e demonstrator, sono controllati da reti diverse. Il lavoro dell'ultimo periodo ha come obiettivo quello di ottenere una situazione in cui i due sono controllati da una stessa rete, in grado di apprendere individualmente, socialmente e fare da demonstrator una volta appreso.

Bibliografia

[1] J.M. Baldwin, *A new factor in evolution*, American Naturalist **30** (1896).

[2] A. Billard and M. J. Mataric, *A biologically inspired robotic model for learning by imitation*, Proceedings of the Fourth International Conference on Autonomous Agents (Barcelona, Catalonia, Spain) (Carles Sierra, Maria Gini, and Jeffrey S. Rosenschein, eds.), ACM Press, 2000, pp. 373–380.

[3] J. Blynel and D. Floreano, *Levels of Dynamics and Adaptive Behavior in Evolutionary Neural Controllers*, 7th International Conference on Simulation on Adaptive Behavior (SAB'2002), 2002, In B. Hallam, D. Floreano, J. Hallam, G. Hayes, and J.-A. Meyer (eds).

[4] J. Blynel and D. Floreano, *Exploring the T-Maze: Evolving Learning-Like Robot Behaviors using CTRNNs*, 2nd European Workshop on Evolutionary Robotics (EvoRob'2003), Lecture Notes in Computer Science, 2003, Raidl, G. et al. (eds.).

[5] E. Borenstein and Ruppin E., *Enhancing autonomous agents evolution with learning by imitation*, AISB Journal (2003).

[6] C. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and B. Blumberg, *Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots*, Artif. Life **11** (2005), no. 1-2, 31–62.

[7] R. A. Brooks, *A robust layered control system for a mobile robot*, Tech. report, Cambridge, MA, USA, 1985.

[8] R. A. Brooks, *Artificial life and real robots*, European Conference on Artificial Life, 1992, pp. 3–10.

[9] P. Cariani, *Some epistemological implications of devices which contruct their own sensor and effectors*, Towards a practice of autonomous systems (Cambridge, MA, USA), MIT Press, 1991, pp. 484–493.

[10] A. Christensen, *Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot*, Tech. Report TR/IRIDIA/2005-014, IRIDIA, Université Libre de Bruxelles, October 2005.

[11] M. Cook and S. Mineka, *Observational conditioning of fear to fear-relevant versus fear-irrelevant stimuli in rhesus monkeys*, Journal of Abnormal Psychology **98** (1989), 448–459.

[12] J. Demiris and G. Hayes, *Imitative learning mechanisms in robots and humans*, 1996.

[13] E. A. Di Paolo, *Evolving spike-timing-dependent plasticity for single-trial learning in robots*, Royal Society of London Philosophical Transactions Series A **361** (2003), 2299–2319.

[14] M. Dorigo and U. Schnepf, *Genetics-based machine learning and behaviour based robotics: A new synthesis*, IEEE Transactions on Systems, Man and Cybernetics **23** (1993), no. 1, 141–154.

[15] D. Floreano and F. Mondada, *Evolution of homing navigation in a real mobile robot.*

[16] D. Floreano and J. Urzelai, *Evolution and learning in Autonomous Robotic Agents*, Bio-inspired Computing Systems (T. Mange and M. Tomassini, eds.), PPUR, Lausanne, 1998.

[17] D. Floreano and J. Urzelai, *Evolutionary Robotics: The Next Generation*, Evolutionary Robotics III (T. Gomi, ed.), AAI Books, Ontario (Canada), 2000.

[18] B.G. jr Galef and S.W. Wigmore, *Transfer of information concerning distant foods: a laboratory investigation of the information-centre hypothesis*, Animal Behavior **31** (1983), 748–758.

[19] Y. Gatsoulis, G. Maistros, Y. Marom, and G. Hayes, *Learning to forage through imitation*, 2003.

[20] I. Harvey, E. Di Paolo, R. Wood, M. Quinn, and E. Tuci, *Evolutionary robotics: A new scientific tool for studying cognition*, Artif. Life **11** (2005), no. 1, 79–98.

[21] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi, *Evolutionary robotics: the sussex approach*, 1996.

[22] N. Jakobi, *Minimal simulations for evolutionary robotics*, 1998.

[23] Y. Kuniyoshi, M. Inaba, and H. Inoue, *Learning by watching: Extracting reusable task knowledge from visual observation of human performance*, T-RA **10** (1994), 799–822.

[24] J. Noble, E. Tuci, and P. M. Todd, *An evolutionary simulation model of social learning about food by norway rats*, European Conference on Artificial Life, 1999, pp. 514–523.

[25] S. Nolfi and D. Floreano, *Learning and evolution*, 1999.

[26] S. Nolfi and D. Floreano, *Evolutionary Robotics. the Biology, Intelligence, and Technology of Self-organizing Machines*, MIT Press, Cambridge, MA, 2001, 2001 (2nd print), 2000 (1st print).

[27] S. Nolfi, O. Miglino, and D. Parisi, *Phenotypic plasticity in evolving neural networks*, 1994.

[28] E. Di Paolo, *Homeostatic adaptation to inversion in the visual field and other sensorimotor disruptions*, 2000.

[29] P. Phattanasri, H. J. Chiel, and R. D. Beer, *The dynamics of associative learning in evolved model circuits*, (2006).

[30] M. Quinn, L. Smith, G. Mayley, and P. Husbands, *Evolving formation movement for a homogeneous multi-robot system: Teamwork and role-allocation with real robots*, Tech. Report 515, School of Cognitive and Computing Sciences, University of Sussex, U.K., 2002.

[31] S.M. Reader, Kendal J.R., and Laland K.N., *Social learning offoraging sites and escape routes in wild trinidadian guppies*, Animal Behavior **66** (2003), 729–739.

[32] A. Thompson, *Evolving electronic robot controller that exploit hardware resources*, European Conference on Artificial Life, 1995, pp. 640–656.

[33] E. Tuci, I. Harvey, and M. Quinn, *Evolving integrated controllers for autonomous learning robots using dynamic neural networks*, 2002.

[34] E. Tuci and M. Quinn, *Behavioural plasticity in autonomous agents: A comparison between two types of controller*, Applications of Evolutionary Computing: EvoWorkshops 2003 (Colchester, Essex) (S. Cagnoni,

J.J.R. Cardalda, D.W. Corne, J. Gottlieb, A. Guillot, E. Hart, C. G. Johnson, E. Marchiori, J-A. Meyer, M. Middendorf, and G.R. Raidl, eds.), vol. 2661, Springer-Verlag, Berlin, 2003, pp. 661–672.

[35] E. Tuci, M. Quinn, and I. Harvey, *An evolutionary ecological approach to the study of learning using a robot based model*, 2003.

[36] T.R. Zentall, *Imitation: definitions, evidence and mechanisms*, Animal Cognition **9** (2006).

Appendice A

Allegato

G. Pini, E. Tuci, and M. Dorigo, *Evolution of social and individual learning in autonomous robots*, European Conference on Artificial Life, 2007.