Adattamento in sistemi embodied & situated

Claudio Martella
Matricola: 686058
Relatore: Prof. N. Alberto BORGHESE
Correlatore: Dott. Stefano NOLFI

Università degli studi di Milano
Facoltà di scienze Naturali, Fisiche e Matematiche
Corso di laurea magistrale in Informatica
A.A 2006/07

Aprile 2008

Indice

1 Introduzione **7**

2 Adattamento in sistemi embodied & situated **9**
2.1 Sistemi embodied and situated 9
2.2 Progettazione e adattamento 10
2.3 Adattamento in sistemi embodied and situated 12
2.4 Il setup sperimentale 15

3 Adattamento evolutivo e individuale **19**
3.1 Introduzione 19
3.2 Simuated annealing 19
3.2.1 Applicazione del SA a sistemi embodied & situated 22
3.2.2 Analisi quantitativa della stocasticità intrinseca nella stima della prestazione 26
3.2.3 Sommario 30
3.3 Algoritmi evolutivi 31
3.3.1 Applicazione di un algoritmo evolutivo al setup sperimentale scelto 33
3.4 Comparazione dei risultati ottenuti 34

4 Adattamento individuale e sociale **37**
4.1 Introduzione 37
4.2 Apprendimento sociale 37
4.3 Apprendimento sociale in sistemi embodied & situated 40
4.3.1 Variazioni al setup sperimentale 43
4.3.2 Applicazione dell'apprendimento sociale al setup sperimentale scelto 44
4.4 Analisi dei risultati 52
4.5 Sommario 54

5 Conclusioni **55**

A Implementazione **57**

Ringraziamenti

Il primo ringraziamento va ai miei genitori che per la n-esima volta mi hanno supportato dall'inizio alla fine. Perchè hanno saputo chiudere un occhio. E perchè hanno saputo chiudere anche l'altro.

Il resto della famiglia la ringrazio per i motivi che mi legano a questa tesi. Non condividiamo solo il ramo dell'albero ma tutta la radice.

Ringrazio Stefano Nolfi per la disponibilità nel seguirmi dal primo giorno al ISTC, con stimoli, correzioni e soprattutto risposte.

Un grazie al prof. Borghese per avermi chiesto di dare di più.

Nuovamente ringrazio i miei amici per i motivi più ovvi, che conoscevano 3 anni fa e spero conoscano tutt'ora.

Grazie a Charles Darwin per avermi dato gli strumenti per provare a comprendere ciò che ci circonda, attraverso l'eleganza che giace nella semplicità.

Rinnovo il mio ringraziamento a Thom Yorke, Ed O'Brien, Colin e Johnny Greenwood e Phil Selway, che ancora non hanno smesso di suonare la loro colonna sonora.

Uno particolare ad Andrea Marcellino (lascio il dubbio su quale sia il nome e quale il cognome) per l'immagine a piè di pagina.

Capitolo 1

Introduzione

Il lavoro in questa tesi si inserisce all'interno del campo dell'intelligenza artificiale, della robotica evolutiva e dei sistemi adattivi.

In questa tesi analizziamo la possibilità di applicare processi adattativi ad agenti embodied e situated prendendo in considerazione algoritmi di apprendimento evolutivo, individuale e sociale. Più precisamente analizzeremo come dei robot mobili provvisti di un controllore neurale possano sviluppare autonomamente, in interazione con l'ambiente, la capacità di esibire semplici comportamenti e di categorizzare oggetti diversi.

Nel capitolo 2 viene analizzato, in termini generali, il problema dell'adattamento in un sistema embodied e situated. Partendo dalle definizioni di embodiement e situatedness introduciamo alcune classi di algoritmi e tecniche di design come gli algoritmi di ottimizzazione iterativa e il divide-and-conquer. Successivamente viene introdotto l'approccio emergente ed auto-organizzante all'adattamento, utilizzato nei restanti capitoli. L'ultima sezione è dedicata alla descrizione del problema in esame e del setup sperimentale utilizzato.

Nel capitolo 3 prendiamo in esame due diversi algoritmi di apprendimento: il Simulated Annealing che può essere utilizzato per implementare una forma di apprendimento individuale e gli algoritmi evolutivi che possono essere utilizzati per realizzare una forma di apprendimento basata su una popolazione di individui. In particolare descriveremo come questi due algoritmi possano essere applicati per sviluppare robot in grado di risolvere lo scenario descritto nel primo capitolo. Inoltre presentiamo e analizziamo delle variazioni di tali algoritmi che risultano essere più efficaci in termini di tempo di calcolo e/o in termini di prestazioni.

Nel capitolo 4 analizziamo la possibilità di sviluppare un algoritmo di apprendimento in grado di sfruttare l'interazione con altri individui soggetti ad apprendimento. In altri termini analizziamo se la possibilità di apprendere in un contesto sociale possa comportare dei vantaggi dal punto di vista algoritmico. In particolare definiremo una funzione per descrivere le differenze comportamentali tra due agenti ed applicheremo l'algoritmo descritto nel capitolo precedente per minimizzare tale funzione obiettivo. Questo algoritmo viene paragonato con l'algoritmo di apprendimento individuale presentato nel capitolo 3 nei termini della performance delle migliori soluzioni ottenute, a parità di costo computazionale, dai due algoritmi.

Capitolo 2

Adattamento in sistemi embodied & situated

2.1 Sistemi embodied and situated

In questo capitolo analizziamo in termini generali la possibilità di applicare processi adattativi ad agenti embodied and situated.

Per agente embodied e situated intendiamo un sistema dotato di un corpo (simulato o reale) posto in un ambiente esterno con il quale interagisce. Sistemi di questo tipo possono risolvere un compito dato esibendo un comportamento che risulta dalla interazione tra il sistema e l'ambiente.[11, 40].

Per processo adattativo intendiamo un processo di apprendimento (basato per es. su algoritmi evolutivi o su simulated annealing) in cui alcune delle caratteristiche dell'agente sono codificate in dei parametri liberi che vengono variati casualmente e in cui le variazioni vengono ritenute o scartate sulla base degli effetti riscontrati al livello del comportamento globale esibito dal sistema (cioè nella misura in cui tali variazioni producono un aumento o un decremento della capacità del sistema di risolvere un compito dato). La modifica dei parametri liberi può avvenire solamente alla "nascita" dell'agente (ed in questo caso si parlerà di adattamento filogenetico, o processo evolutivo) oppure durante la "vita" dello stesso (ed in questo caso si parlerà di adattamento ontogenetico, o processo di apprendimento o sviluppo).

L'agente, nelle sue parti, e l'ambiente sono caratterizzati dalle loro caratteristiche fisiche soggette alle leggi della fisica (reale o simulata che sia) e alle interazioni con l'ambiente in tutte le sue forme. I dati sensoriali sono spesso incompleti e soggetti a rumore, locali al soggetto e contestuali al suo punto di vista.

In questo contesto non si tratta dello sviluppo di un sistema di controllo esternamente e dell'incorporamento dello stesso a posteriori, ma dell'emergenza del sistema stesso attraverso l'interazione dell'agente con l'ambiente (interno ed esterno). Va preso in considerazione il fatto che i dati sensoriali non sono solo il risultato della posizione dell'agente nell'ambiente ma anche delle azioni eseguite che l'hanno portato in quello stato ed hanno quindi determinato, attraverso l'interazione agente-ambiente, i valori sensoriali. Si crea in questo modo

una relazione che lega lo stato in cui si trovano l'agente e l'ambiente, l'azione intrapresa e lo stato successivo.

Queste non sono tuttavia da considerarsi solamente come delle limitazioni ma anche come delle caratteristiche che il processo può sfruttare per far emergere dei comportamenti robusti di interazione tra l'agente e l'ambiente. Si può arrivare quindi a definire l'embodiement (definizione forte) come la capacità dell'agente di sfruttare le caratteristiche del suo corpo per risolvere il problema adattivo e situatedness (definizione forte) come la capacità di sfruttare le interazioni possibili con l'ambiente per la risoluzione del problema adattivo.

2.2 Progettazione e adattamento

Nel campo dell'ottimizzazione esistono vari metodi per risolvere problemi NP-complete (problemi la cui complessità cresce in modo più che polinomiale al crescere del numero di elementi), attraverso diverse strategie di euristica.

Una strategia è il "divide-and-conquer" in cui il problema viene diviso in sottoproblemi più facili da risolvere ed una volta trovata una soluzione per ognuno di essi queste vengono ricomposte. Ovviamente per poter applicare questa strategia è necessario che i sottoproblemi siano naturalmente disgiunti.

Un'altra strategia consiste nell'ottimizzazione iterativa in cui un sistema parte da uno stato noto e delle modifiche al sistema vengono iterativamente applicate in modo da minimizzare una funzione costo. Dal momento che questo approccio porta spesso, per definizione, ad un problema di minimi locali normalmente l'ottimizzazione iterativa viene applicata più volte con configurazioni iniziali differenti e salvando il miglior risultato.

Il paradigma di programmazione "divide-and-conquer" prevede che un problema possa essere diviso ricorsivamente in sottoproblemi finchè questi non diventino semplici da risolvere. Le soluzioni ai sottoproblemi vengono poi ricomposte nella soluzione al problema originale. Questo tipo di paradigma non si discosta molto dalla relazione ricorsiva che lega alcune funzioni. Spesso questo tipo di funzioni sono infatti implementate tramite funzioni ricorsive, oppure tramite funzioni non ricorsive ma con l'ausilio di strutture dati come pile, code, etc. Alcuni esempi di algoritmi noti che si basano su questo paradigma sono il quicksort e mergesort.

Questo tipo di tecnica può essere utilizzata in ambiti in cui la divisione in sottoproblemi è concettualmente chiara e diretta o addirittura quando essa è l'unica soluzione semplice (ad esempio come il problema della torre d'hanoi o gli algoritmi di ordinamento e ricerca). Nel caso di un sistema embodied e situated, dove l'enfasi viene posta sul comportamento, la disgiunzione perde la naturalezza propria di problemi puramente logico-matematici. In questo caso non è possibile dividere l'agente dall'ambiente ed il comportamento esibito è il frutto del flusso continuo di interazioni agente-ambiente ed è quindi difficile non solo concettualizzare i vari moduli in gioco ma anche l'interazione tra gli stessi durante questo flusso continuo.

Anche potendo dividere il problema in sottoproblemi e progettare la loro interazione nelle varie condizioni ambientali, permettere all'algoritmo di eseguire tale divisione in sottoproblemi e trovare la loro implementazione permette al sistema di trovare soluzioni originali che lo sperimentatore potrebbe non aver immaginato e soprattutto che siano intrinsecamente efficienti, oltre che capaci

di generalizzare le condizioni dell'ambiente, inquanto emergenti dall'interazione stessa. Per questo motivo per "divide-and-conquer" in questo ambito si intende più un'azione generale di divisione in sottogruppi non formalmente distinti eseguita dall'algoritmo piuttosto di una condizione di progettazione rigorosa dall'alto verso il basso.

L'applicazione di un algoritmo di ottimizzazione iterativa prevede la definizione di un insieme di configurazione, di una funzione costo e di una funzione di generazione, ad es. una funzione che con delle perturbazioni costituisca una transizione da una configurazione ad un'altra.

Questa funzione di generazione crea un insieme Ri per ogni configurazione i di configurazioni vicine, che possono cioè essere raggiunte da i con una sola transizione. Per questo motivo l'algoritmo di ricerca iterativa è considerato un algoritmo di "neighbourhood search" o "local search" [31]. L'algoritmo può quindi ora essere definito come segue. Partendo da una configurazione data (spesso casuale), viene generata una sequenza di iterazioni, ogniuna di esse consistente in una possibile transizione dalla configurazione attuale ad una selezionata dall'insieme delle configurazioni vicine (neighbourhood). Se la configurazione vicina ha un valore di costo minore di quella attuale la transizione verrà accetta e la nuova configurazione diventerà la configurazione corrente, altrimenti un'altra transizione vicina verrà valuta. L'algoritmo termina quando nessuna configurazione vicina presenta un costo minore.

Gli svantaggi di questo tipo di approccio sono:

1. per difinizione gli algoritmi iterativi terminano in un minimo locale e generalmente non c'è informazione sulla relazione tra questo minimo locale e quello globale.

2. il minimo locale ottenuto dipende dalla configurazione di partenza per la quale non esiste un criterio di scelta generale

3. generalmente è impossibile definire un limite superiore di costo computazione

Ciò nonstante questo tipo di tecnica presenta molti fattori vantaggiosi: è applicabile in modo generale, la funzione costo e il meccanismo di generazione sono solitamente facilmente definibili.

Alcune delle possibili soluzioni a questi svantaggi sono:

1. eseguire l'algoritmo per un grande insieme di configurazioni iniziali, nonostante il costo computazionale

2. utilizzare le informazioni sui minimi locali trovati per scegliere la configurazione iniziale delle esecuzioni successive

3. sviluppare di più affinati meccanismi di generazione delle transizioni vicine per permettere all'algoritmo di saltare fuori dal minimo locale

4. accettare con un limitato criterio transizioni con costo maggiore

La seconda e la terza soluzione sono solitamente dipendenti dal problema. La prima soluzione è una soluzione sempre valida che viene applicata generalmente agli algoritmi di ottimizzazione combinatoria. Il quarto approccio è quello utilizzato dal Simulated Annealing[39].

L'algoritmo iterativo può essere utilizzato sia per sviluppare completamente la soluzione del problema sia per sviluppare o ottimizzare le sottosoluzioni ai sottocomportamenti definiti dallo sperimentatore. In questo modo si può applicare un approccio ibrido in cui lo sperimentatore divide il problema in sottoproblemi e definisce la ricomposizione delle sottosoluzioni, in stile divide-and-conquer, lasciando che sia l'ottimizzazione iterativa a trovarle.

2.3 Adattamento in sistemi embodied and situated

Gli algoritmi di addattamento in sistemi embodied and situated presentano delle caratteristiche auto-organizzanti che prevedono un ulteriore passo rispetto alla soluzione presentata precedentemente che prevede la suddivisione in sottoproblemi da parte dello sperimentatore e la ricerca delle sottosoluzioni da parte dell'ottimizzazione iterativa.

Si immagini uno scenario in cui un agente debba risolvere un problema di discriminizzazione di oggetti disseminati per l'ambiente come buoni o cattivi. Una soluzione come quella ibrida prevederebbe una suddivisione del sistema di controllo ad esempio in tre moduli: quello di esplorazione, quello di discriminazione e quello di approccio/evitamento dell'oggetto. I moduli verrebbero così sviluppati dal algoritmo di ottimizzazione iterativa ed allo sperimentatore non rimarrebbe che sviluppare i moduli che garantiscano la loro interazione e collaborazione.

Tuttavia, l'applicazione di algoritmi adattativi ad agenti embodied and situated presenta alcune caratteristiche peculiari.

Un primo aspetto da considerare è costituito dal fatto che il comportamento di questi sistemi è il risultato delle interazioni tra il sistema di controllo, il corpo dell'agente, e l'ambiente e non può essere ricondotto a nessun di questi tre elementi in isolamento [6, 28, 7]. Ciò implica che, dal punto di vista dello sperimentatore, può risultare difficile stabilire come il sistema deve comportarsi in ciascuna possibile situazione allo scopo di produrre un comportamento che risolva il compito dato. Tutto questo rende poco praticabile l'utilizzo di tecniche di apprendimento supervisionate in cui lo sperimentatore specifica la risposta desiderata che il sistema deve fornire in ciascuna possibile situazione.

Un approccio completamente adattivo al problema prevede che sia il sistema stesso a scegliere se e come dividere il problema in sottoproblemi e come ricomporre le soluzioni. Esso può scoprire differenti modi di dividere il comportamento di alto livello richiesto dalla soluzione in sottocomportamenti che sfruttino le caratteristiche dell'agente, dell'ambiente e delle possibili interazioni tra questi che siano robusti ed efficienti, nei termini delle risorse a disposizione.

Ad esempio in [16] gli autori evolvono il sistema di controllo di un gruppo di robots assemblati in una struttura lineare per un comportamento coordinato di approccio motorio alla fonte di luce. I robots sono forniti di una torretta, che ruota sull'asse verticale grazie ad un motore, collocata sopra lo chassis. La torretta è fornita di un giunto, comandato da un altro motore, che lo collega ad un altro robot. Ogni robot è inoltre fornito di un sensore di luce e di un sensore di trazione che riporta i valori di intensità e direzione della trazione esercitata sullo chassis. Dato che non sempre la luce è visibile dai robot da

tutte le posizioni e che i robot inizialmente possono partire con orientamenti differenti, essi devono esibire un comportamento di coordinazione per muoversi tutti verso la fonte di luce non appena uno di essi ne rilevi la presenza.

I robots esibiscono un comportamento di negoziazione e coordinazione per approcciare la fonte luminosa non appena il gradiente luminoso viene rilevato. Testando questi sistemi di controllo in diverse condizioni, essi esibiscono la capacità di generalizzare il problema esibendo anche comportamenti nuovi non descritti esplicitamente nella fitness.

Analizzando il comportamento esibito gli autori riscontrano i seguenti sottocomportamenti: un comportameto di avanzamento (utilizzato quando in coordinamento con gli altri robots, lontano dagli ostacoli e rilevato il gradiente di luce), un comportamento conformista (l'abilità del robot di adeguare il suo orientamento a quello del team nel momento in cui questi siano molto diversi), un comportamento di fototassi (la capacità dei robots di orientarsi verso la fonte luminosa), un comportamento di evitamento degli ostacoli. L'interazione tra questi sottocomportamenti genera un gruppo di comportamenti di più alto livello: un comportamento di movimento coordinato (un comportamento coordinato che permette al team di raggiungere la destinazione ovviando a disallienamenti generati man mano nel movimento), un comportamento coordinato di approccio alla luce e un comportamento coordinato di evitamento degli ostacoli. L'interazione tra questi porta ai seguenti comportamenti di livello ancora più alto: un comportamento collettivo di esplorazione e un comportamento di riarrangiamento della disposizione spaziale del gruppo nell'arena (per passare ad esempio attraverso zone dell'arena come vicoli stretti).

Man mano che si passa dai sottocomportamenti ai livelli superiori il comportamento risultante ha una durata sempre maggiore rispetto ai sottocomportamenti che lo "compongono" essendo il risultato di multiple interazioni tra gli stessi. Questo tipo di suddivisione in diversi livelli di sottocomportamenti e la loro interazione è stata generata dall'algoritmo evolutivo, senza l'intervento diretto dello sperimentatore che ha definito solo tre regole di controllo nella funzione di fitness: la regola che definisce il movimento diretto quando non vi è trazione sulla torretta e non vi è luce ai lati del robot, quella che regola il movimento verso la direzione della trazione sulla torretta, quando questa è elevata e quella che regola l'orientamento del robot verso la fonte di luce quando questa è intensa. L'esibizione di tutti i sottocomportamenti descritti, in un lasso di tempo, porta all'esibizione del comportamento completo richiesto, e cioè l'abilità esibita dai robots di assemblarsi ed esplorare l'arena in modo coordinato approcciando la fonte luminosa ed evitando gli ostacoli.

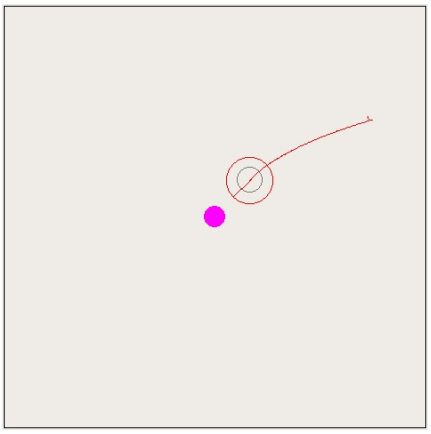
Sistemi di questo tipo possono essere sviluppati adeguatamente attraverso l'utilizzo di tecniche adattive evoluzionistiche in cui i parametri liberi del sistema regolano le interazioni fini tra il sistema e l'ambiente e in cui le variazioni di tali parametri vengono ritenute o scartate sulla base degli effetti al livello del comportamento globale esibito dal sistema (o più precisamente sulla base della capacità del sistema di risolvere il compito dato).

Un secondo aspetto da considerare è il fatto che i sistemi embodied and situated sono sistemi dinamici in cui piccole variazioni nelle condizioni iniziali (per es. nella posizione iniziale dei robot) o piccole variazioni dovute ad effetti stocastici (per es. il rumore sui sensori e sugli effetti degli attuatori) possono produrre effetti significativi. Tale fatto implica l'impossibilità di ottenere una misura precisa delle prestazioni del sistema. La misura delle prestazioni del si-

stema dunque rappresenta solo una stima delle prestazioni effettive calcolabili, in linea di principio, mediando le prestazioni stimate su un numero infinito di prove. Il costo computazionale limita il numero di volte che una soluzione può essere valutata rendendo la stima delle prestazioni più rumorosa (*horizon problem*). Per questo motivo il numero di epoche, cioè il numero di volte che una soluzione viene valutata a partire da uno stato iniziale differente, è spesso un parametro molto importante del processo. Un numero di valutazioni troppo basse vanificherebbe l'utilità della funzione di fitness, dal momento che la valutazione risulterebbe troppo rumorosa rendendo difficile l'emergenza dei comportamenti più promettenti. Dall'altro lato la funzione di valutazione è spesso il componente più dispendioso in termini di costo computazionale e lo sperimentatore vorrà quindi evitare di utilizzarla più del necessario.

In questi sistemi si utilizza una funzione detta funzione obiettivo che descrive l'efficacia di una soluzione, detta fitness. Questa deve contenere sufficiente informazione da permettere al processo di selezionare le soluzioni e le sottosoluzioni più promettenti sia nei termini della performance totale che nei termini del dispendio di risorse. Quest ultimo livello di ottimizzazione può avvenire in due modi: specificando esplicitamente i costi nella funzione obiettivo (chiedendo quindi di minimizzarli) oppure più implicitamente attraverso lo sfruttamento delle caratteristi dell'agente e dell'ambiente. La funzione obiettivo deve essere tuttavia sufficientmente generale da non spingere il processo in una direzione prevalicandone altre, evitando di specificare dettagliatamente le caratteristiche che le soluzioni devono presentare.

(a) E-puck



(b) Evorobot

Figura 2.1: Robot reale e simulato

2.4 Il setup sperimentale

Il robot è costituito dal robot mobile e-puck sviluppato al Politecnico di Losanna. Gli esperimenti sono effettuati in simulazione utilizzando una versione estesa del software Evorobot sviluppato da Stefano Nolfi e collaboratori all'Istituto di Scienze e Tecnologie della Cognizione del CNR di Roma (entrambi in figura 2.1).

Il robot viene lasciato "vivere" all'interno di un'arena quadrata di 500cmx500cm. All'inizio di ogni prova, un elemento di cibo viene posizionato al centro dell'arena mentre il robot viene posizionato in un punto casuale della stessa. I 10 elementi di cibo sono divisi in in 2 gruppi, 5 che vengono classificati come buoni e 5 come cattivi. Ogni elemento è caratterizzato da un colore all'interno dello spazio RGB.

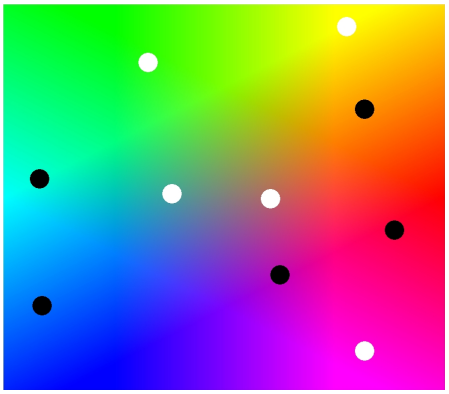
La codifica dei colori degli oggetti (figura 2.2) è stata selezionata accuratamente al fine di distribuire gli oggetti all'interno dello spazio colore di modo da coprire maggiore superficie possibile dello stesso e mantenendo la massima distanza possibile tra gli elementi evitando che oggetti con la stessa caratteristica finiscano vicini nella codifica.

Il robot è fornito di 8 sensori infrarossi in grado di rilevare la presenza di ostacoli, una telecamera omnidirezionale in grado di rilevare il colore e la posizione del'oggetto. Il robot e' inoltre fornito di due motori che controllano la velocita' desiderata delle due ruote corrispondenti.

Il sistema di controllo di ogni robot è costituito da una rete neurale che include 13 neuroni di input, 2 neuroni nel layer nascosto e 2 neuroni di output. I neuroni di input sono collegati ai neuroni nascosti e direttamente ai neuroni di output.

I primi 8 neuroni di input codificano i valori degli 8 sensori IR, i neuroni 9, 10 ed 11 rispettivamente i valori RGB dell'oggetto mentre il 12 ed il 13 codificano la posizione (sinistra e destra) dell'elemento di cibo relativa al centro del campo visivo del robot con un segnale di intensità proporzionale alla distanza dal centro del campo visivo.

I parametri liberi del sistema che vengono modificati durante il processi di



Buoni	Cattivi
61, 208, 208	27, 243, 17
15, 86, 174	11, 148, 146
254, 194, 62	234, 140, 234
208, 98, 194	187, 56, 89
223, 77, 98	247, 238, 45

Figura 2.2: Gli oggetti sullo spazione colore e la codifica. I punti bianchi indicano oggetti cattivi mentre i punti neri indicano gli oggetti buoni. Si noti che non vi è perfetta corrispondenza tra le distanze tra i punti nello spazio tridimensionale RGB e quello bidimensionale rappresentato in figura.

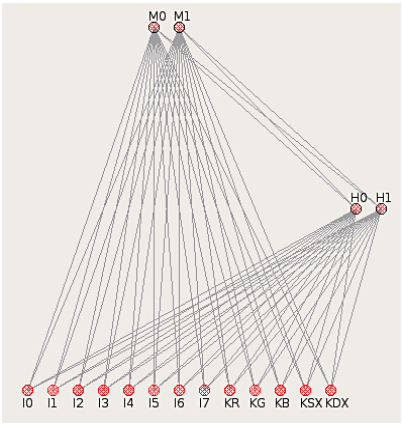


Figura 2.3: L'architettura della rete neurale.

adattamento sono costituiti dai pesi della rete neurale (56 pesi e 4 biases per un totale di 60 parametri). Ciascun parametro è codificato con 8 bit e normalizzato nell'intervallo [-5.0, 5.0].

Ad ogni istante di tempo viene passato alla rete neurale un vettore di numeri reali che descrivono l'output dei sensori corrispondenti agli stessi. Nel caso dei sensori IR ad esempio verrà passato un valore corrispondente alla distanza del robot dagli ostacoli all'interno del range di distanza del quale il sensore lavora. Come risultato la rete neurale fornirà la tensione da applicare ai 2 motori.

Nella parte destra di figura 2.4 si può notare l'attività neuronale. Sull'asse orizzontale si trova il tempo t mentre sull'asse verticale l'attivazione del neurone. I primi due neuroni indicano l'output, i secondi due i neuroni del layer nascosto mentre i restanti stanno ad indicare l'attività dei neuroni di input.

Tracciando una linea verticale in qualunque punto t di questi grafici si può ottenere il vettore di input ed il vettore di output del sistema di controllo in quell'istante di tempo.

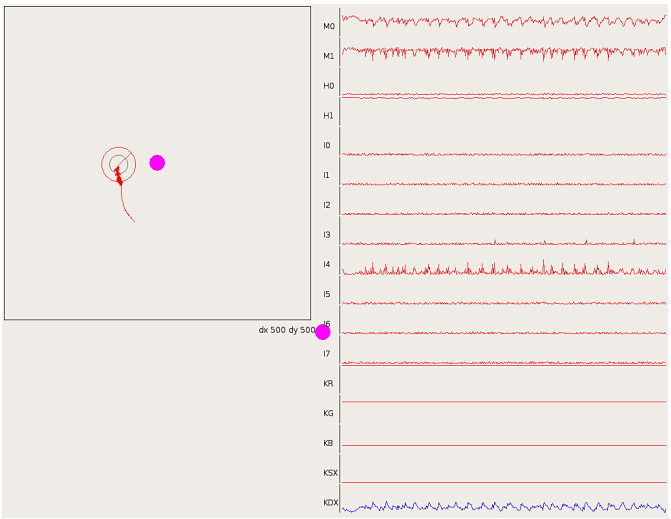


Figura 2.4: Il robot nell'arena e l'attività dei neuroni ad ogni istante.

Il compito che il robot deve svolgere consiste nel raggiungere e rimanere vicino agli oggetti buoni e allontanarsi e rimanere lontano dagli oggetti cattivi raggiungendo una distanza inferiore a superiore ai 150mm, rispettivamente. Le prestazioni del robot vengono valutate per un certo numero di epoche. Ogni epoca è costituita da 10 prove corrispondenti ai 10 oggetti di colore diverso che l'individuo deve categorizzare. Ogni prova è costituita da 300 passi di 100ms ciascuno. Alla fine di ciascuna prova il robot riceve un punteggio di 1 ogni volta che la posizione del robot rispetta il criterio descritto sopra e un punteggio di -1 in caso contrario. Il robot riceve un punteggio di -1 anche nel caso in cui entri in collisione con l'oggetto o con le mura. La fitness del robot corrisponde dunque al punteggio complessivo normalizzato rispetto al numero di epoche e di prove. Per semplicità la fitness viene poi traslata dall'intervallo $[-1, 1]$ a $[0, 2]$.

Questo setup sperimentale presenta una serie di caratteristiche che lo rendono adatto allo scopo che ci si prefigge in questa tesi.

Il primo aspetto da considerare è costituto dal fatto che si tratta di un esperimento "embodied" e "situated" in cui il sistema (il robot) sviluppa una capacità in interazione con l'ambiente esterno nel quale e' situato. Questo aspetto ha delle importanti implicazioni che riguardano, per es., il fatto che le esperienze vissute durante la fase di adattamento sono una funzione del modo in cui il robot reagisce agli stimoli ambientali.

Un secondo aspetto da considerare è costituito dal fatto che la funzione di valutazione che guida il processo di adattamento fornisce una valutazione complessiva del comportamento complessivo esibito dal robot. Tale funzione non fornisce dunque alcuna indicazione specifica su come il robot deve reagire ad un dato stimolo sensoriale. Questo aspetto, a sua volta, presenta vantaggi e svantaggi. Il vantaggio è costiuito dal fatto che il robot è libero di selezionare la strategia con cui risolvere in compito (per es. la traiettoria con cui si avvicina o si allontana da un oggetto). Lo svantaggio è costituito dal fatto che l'informazione limitata fornita dalla funzione di valutazione può essere insufficiente per guidare il processo di adattamento nella direzione giusta. Questo aspetto può

essere illustrato nel contesto di questo setup sperimentale facendo riferimento al fatto che ragionevolmente i robot devono sviluppare due capacità interdipendenti. Una capacità di categorizzare gli oggetti di diverso colore in due categorie (oggetti che devono essere raggiunti e oggetti che devono essere evitati) e una capacità di avvicinarsi o allontanarsi dall'oggetto corrente. Il fatto che la funzione di valutazione tiene in conto solo il comportamento complessivo esibito dal robot (cioè la capacità di raggiungere o evitare gli oggetti a seconda della loro categoria) e non anche la capacità di categorizzare correttamente gli oggetti (a prescindere dalla capacità di reagire nel modo appropriato) può causare il fatto che variazioni dei parametri liberi che producono un miglioramento della capacità di categorizzare, la quale tuttavia non produce a sua volta un aumento della capacità di avvicinarsi o allontanarsi dall'oggetto, non siano ritenute.

In generale dunque si può osservare che l'utilizzo di funzioni di valutazioni che tengono conto solo del comportamento complessivo del robot e che non rinforzano direttamente lo sviluppo delle capacità più elementari che sono necessarie per esibire il comportamento complessivo desiderato può rallentare o impedire il successo del processo di adattamento.

Un terzo aspetto da considerare è costituito dal fatto che in questo setup sperimentale i robot devono essere in grado di trattare in modo appropriato un numero significativo di diverse situazioni ambientali (costituite da oggetti di colore diverso che richiedono risposte diverse). Poiché la relazione tra il colore degli oggetti e il comportamento appropriato corrispondente è arbitraria, la soluzione del problema richiede lo sviluppo di una serie di capacità interdipendenti.

Sarebbe stato possibile individuare le capacità ed i sottocomportamenti necessari per la risoluzione del problema generale e richiedere all'algoritmo adattivo di svilupparli specificando ognuno di essi nella funzione obiettivo. Questo approccio, descritto nel paragrafo 2.3, prevede di massimizzare la funzione obiettivo facendo emergere le sottosoluzioni in essa specificate. Volendo ipoteticamente implementare tale funzione di fitness, essa sarebbe composta da più componenti. Una componente richiederebbe all'agente di categorizzare correttamente l'oggetto sulla base dei dati sensoriali proveniente dai neuroni RGB mentre un'altra componente richiederebbe di esplorare l'arena evitando gli ostacoli ed i muri. Una tera componente richiederebbe all'agente di rimanere vicino agli oggetti buoni e lontano da quelli cattivi. L'algoritmo adattivo implementerebbe le strategie per mettere in atto queste soluzioni definendo ad esempio il modo in cui l'oggetto nello specifico debba comportarsi di fronte agli input sensoriali dell'ambiente nelle diverse condizioni (ad esempio con che angolo ed intensità avvicinarsi o allontanarsi dall'oggetto, la traiettoria utilizzata per esplorare l'arena, la distanza a cui tenersi dai muri etc.).

La soluzione adottata in questa tesi, come specificato precedentemente, è stata di utilizzare una funzione che premia l'agente sulla base della distanza tra esso e l'oggetto a seconda delle sue caratteristiche (lontano se l'oggetto è cattivo e vicino viceversa). Essa inoltre viene calcolata solamente alla fine del trial e non colleziona dati durante lo stesso. Nella funzione non viene specificata ne la necessità di categorizzare gli oggetti ne l'uso dei neuroni RGB, non viene chiesto di stare lontano dall'oggetto durante il trial (ad esempio calcolando la distanza ad ogni istante del trial) e non viene chiesto esplicitamente di evitare gli ostacoli (una collisione viene trattata come una categorizzazione errata).

Il comportamento sviluppato dall'algoritmo adattivo è quindi emergente ed auto-organizzante.

Capitolo 3

Adattamento evolutivo e individuale

3.1 Introduzione

In questo capitolo prenderemo in esame due diversi algoritmi di apprendimento: il Simulated Annealing che può essere utilizzato per implementare una forma di apprendimento basato su un singolo individuo e gli algoritmi evolutivi che possano essere utilizzati per realizzare una forma di apprendimento basata su una popolazione di individui. In particolare descriveremo come questi due algoritmi possano essere applicati per sviluppare robot in grado di risolvere lo scenario descritto nella sezione 2.4. Inoltre dimostreremo come alcune variati di tali algoritmi sperimentati dall'autore di questa tesi si dimostrino più efficaci almeno nel campo di applicazione scelto. Infine compareremo i risultati ottenuti con i due tipi di algoritmo.

3.2 Simuated annealing

Il Simulated Annealing [22] è un algoritmo utilizzato nel campo dell'ottimizzazione e mima un processo noto come "annealing" utilizzato in metallurgia. Nel capo dell'ottimizzazione l'algoritmo consiste in un ciclo di transizioni che avvengono da uno stato ad uno vicino. Queste transizioni vengono accettate qualora queste migliorino l'energia del sistema ma anche, con una probabilità P, qualora queste la peggiorino. Questa probabilità P è un valore variabile che, inizialmente alto, scende exponenzialmente: $exp(\frac{(b-a)}{T})$ dove a è l'energia prima della transizione e b dopo la stessa, mentre T è un parametro che diminuisce con il progredire del processo di ottimizzazione.

La temperatura così modellata permette al SA, durante la fase iniziale del processo di adattamento, di esplorare soluzioni diverse evitando una convergenza prematura a soluzioni subottimali e di ottimizzare la soluzione trovata durante la fase finale.

Il seguente pseudo-codice descrive l'algoritmo standard[22]:

```
c := 0
curr_perf := 0
curr_sol  := random_solution()
while c < learning_cycles:
    T := ((learning_cycles - c)/learning_cycles) * starting_T
    new_sol  := mutate(curr_sol)
    new_perf := test(new_sol)
    if new_perf >= curr_perf OR rand() <= exp((new_perf - curr_perf)/T):
        curr_sol  := new_sol
        curr_perf := new_perf
    c := c + 1
```

Un esempio di applicazione del Simulated Annealing è il problema del commesso viaggiatore [22]. Per cominciare si può pensare di codificare le città con dei numeri interi. Un itinerario potrà essere costituito da una stringa di questi numeri che definisce, nell'ordine, le città che verranno visitate. Tutte le possibili permutazioni di questi numeri costituiscono lo spazio delle soluzioni. Con una funzione si può calcolare il costo della soluzione attraverso il calcolo delle distanze tra tutte le diverse città, nell'ordine. Dato un itinerario di N siti, la funzione che calcola il costo di itinerario sarà $E = \sum_{i=2}^{N} d_{i-1,i}$ dove $d_{i-1,i}$ rappresenta la distanza tra il sito i e $i-1$. Un'altra funzione necessaria è quella che applica le modifiche ad una soluzione per cercarne una migliore.

Normalmente esistono delle strategie per creare una nuova soluzione a partire da un'altra, nel caso del simulated annealing una mutazione casuale può essere una strategia molto efficace di esplorazione.

Dal momento che l'algoritmo deve trovare la sequenza di città, codificate con degli interi, che dovranno essere visitate senza ripetizioni dal commesso viaggiatore con il minimo costo, l'algoritmo di generazione delle soluzioni attraverso mutazioni non può essere un operatore completamente casuale dal momento che potrebbe generare combinazioni invece di permutazioni, oppure soluzioni illegali, cioè soluzioni che non rappresentano un itinerario possibile (ad esempio perché due città vicine nell'itinerario non sono collegate fisicamente). Anche se non applicabile al SA ma solo agli algoritmi genetici basati su popolazione, ha inoltre senso utilizzare il crossover: la ricombinazione di sottopercorsi all'interno di un percorso più grande può essere sensato in un contesto di ottimizzazione di questo tipo, nel quale i parametri non descrivono un comportamento come nel caso dei pesi di una rete neurale. Tuttavia, come già sottolineato, l'operatore di crossover deve essere specifico al problema, evitando ripetizioni all'interno del percorso ma generando soluzioni legali.

Quando l'algoritmo venne applicato per la prima volta a questo tipo di problema, esso fu utilizzato efficacemente per trovare una soluzione ad un problema con 6000 siti circa da visitre. La soluzione migliore ottenuta fino a quel momento presentava una dimensione di soli 318 siti [22].

In [37] il SA è stato utilizzato per la ricostruzione di immagini PET (positron emition tomography) da dati di proiezione rumorosi. La PET è una metodologia per catturare su slide l'attività metabolica del corpo di un paziente passando per esso. Solitamente questo processo porta a dei dati rumorosi che devono essere quindi ricostruiti. Il SA è un buon algoritmo per questo tipo di problema per via della sua capacità di non rimanere bloccato in minimi locali. Il metodo alla base

consiste nella generazione iterata di un'immagine che corrisponda ai dati raccolti P^m. Ad ogni iterazione una pseudoimmagine P^p viene generata paragonata ai dati misurati P^m e se la distanza tra questa e la pseudoimmagine ottenuta finora è minore, questa diventerà la nuova immagine corrente, convergendo lentamente verso l'immagine originale.

La funzione che calcola la distanza tra la pseudoimmagine ed i dati raccolti funziona a livello di differenze tra i pixels che compongono le due immagini $E = \sum_{i,j}(P^m_{ij} - P^p_{ij})^2$ (dove i e j indirizzano i pixel). Il meccanismo di generazione consiste in una modifica casuale ad uno o più pixels nella configurazione attuale.

SA viene comunemente utilizzato per la progettazione di circuiti IC [22, 34]. Il problema consiste nel dividere i gates di un diagramma logico tra più chips e posizionarvi i circuiti che compongono i gates minimizzando lo scambio di segnali tra chips e massimizzando il bilanciamento logicistico tra gli stessi. La funzione di fitness calcolerà questi due criteri mentre il meccanismo di generazione invertirà la posizione dei circuiti tra i due chips casualmente.

In [24] viene descritto un utilizzo del SA per la pianificazione della traiettoria che un robot deve eseguire per raggiungere un target evitando degli ostacoli. Al sistema vengono dati il punto di partenza e quello di destinazione più un numero fisso di punti di controllo per definire la *B-spline* che permetta di raggiungere il punto di destinazione con una traiettoria che sia più "smooth" possibile e che minimizzi la distanza percorsa. Le quattro condizioni che devono essere soddisfatte sono il raggiro di un poligono convesso che definisce il perimetro dell'ostacolo, la minimizzazione del percorso totale, il mantenimento delle relazioni parametriche e spaziali costanti lungo il percorso ed il mantenimento delle curvature il più piccolo possibile. Queste quattro condizioni andranno a definire la funzione costo.

Il meccanismo di generazione consiste nella variazione della posizione dei punti di controllo attraverso 4 variabili aleatorie per definire la direzione di spostamento sui 2 assi e la quantità dello stesso. La dimensione delle variazioni è dipendente dalla temperatura e quindi grandi variazioni avvengono solo nella prima fase del processo di apprendimento.

Si noti come in questo lavoro l'algoritmo di apprendimento non viene utilizzato per addestrare il sistema di controllo del robot che determina come il robot reagisce ai diversi stimoli sensoriali (così come ci proponiamo in questa tesi) ma solo per determinare la traiettoria che deve essere esibita dal robot.

Da questi due ultimi esempi emerge la doppia natura del SA. Infatti in entrambi i lavori alle alte temperature iniziali il SA trova soluzioni generali grossolane che vengono affinate a basse temperature. Questo è un esempio di

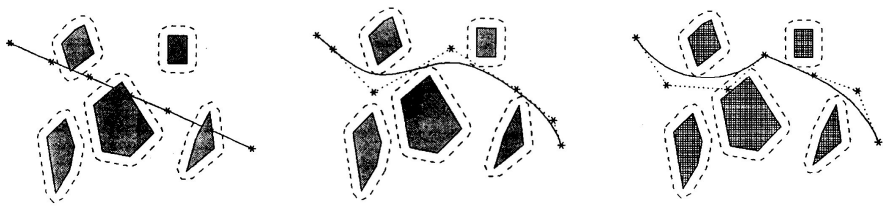


Figura 3.1: Sintesi di B-spline nel problema di pianificazione di traiettorie. Configurazione iniziale, ottimizzata per "smoothness", ottimizzata con discontinuità definita dall'utente.

come un algoritmo di origine iterativa come il SA si comporti anche come un algoritmo divide-and-conquer. Viene inoltre riportato che selezionando migliori condizioni iniziali viene influenzato solamente la velocità di convergenza alla soluzione ottima senza però progiudicare il ritrovamento della stessa.

In[23] gli autori riportano molteplici esempi di utilizzo di SA in ambiti di ricerca operativa.

Sebbene definire la funzione costo ed il meccanismo di generazione non sia particolarmente complesso è comunemente condiviso che l'algoritmo soffra della mancanza di un buon criterio per la scelta dei parametri del sistema come la temperatura iniziale. Per questo, gli ultimi anni hanno visto il proliferare di diverse versioni del SA originale che introducono nuovi criteri di accettazione delle modifiche che portano il sistema a configurazioni a costi maggiori. Uno di questi è il Thermodynamic Simulated Annealing, lo scheduling dell'annealing varia seguendo le leggi della termodinamica, il Adaptive Simulated Annealing o il Very Fast Simulated re-Annealing dove i passi ad ogni iterazione sui parametri sono random, controllati dallo stato del processo, di modo che lo scheduling non debba essere definito direttamente dallo sperimentatore.

3.2.1 Applicazione del SA a sistemi embodied & situated

In questa sezione analizzeremo come il SA possa essere utilizzato per consentire ad un sistema embodied e situated di sviluppare la capacità di risolvere un compito dato. Ciò verrà illustrato nel contesto dello scenario sperimentale illustrato nella sezione 2.4. Nel fare questo discuteremo il problema della stocasticità intrinseca insita nel processo di valutazione della prestazione e le implicazioni di tale stocasticità dal punto di vista algoritmico.

Un primo aspetto da osservare a tale riguardo che, da ciò che risulta all'autore, non esistono in letteratura esempi di utilizzo del SA per addestrare il sistema di controllo di un robot che deve svolgere un compito dato (escluso un recente lavoro di Acerbi e Nolfi[2]). Gli esperimenti descritti in questa sezione dunque rappresentano uno dei primi tentativi in questa direzione.

Un secondo aspetto da considerare è costituito dal fatto che l'applicazione a questo tipo di problemi coinvolge un aspetto che è di norma non rilevante o meno rilevante in altri tipi di applicazioni. Tale aspetto consiste nel fatto che la prestazione del robot rispetto al compito dato può essere fortemente dipendente dalla posizione iniziale dei robot (che è di norma assegnata in modo casuale) e/o dalle caratteristiche dell'ambiente (per es. dalla posizione degli ostacoli). Questo fatto implica che il calcolo della prestazione del robot rappresenta una stima della prestazione ideale del robot (calcolabile in linea di principio testando il robot per un numero infinito di volte al variare della posizione del robot stesso e delle caratteristiche ambientali) e non una misura precisa. Tale stima introduce un ulteriore elemento di stocasticità che si somma alla stocasticità introdotta attraverso la possibilità di accettare variazioni non adattive e contro-adattive regolata della temperatura.

Queste osservazioni suggeriscono come l'algoritmo standard descritto sopra e utilizzato anche da Acerbi e Nolfi nella loro applicazione ad un sistema embodied e situated potrebbe non rivelarsi ottimale per l'applicazione a sistemi, come i sistemi embodied e situated, in cui la funzione di valutazione è in grado di fornire una stima della prestazione e non una misura accurata. Per questa ragione descriveremo qui di seguito i risultati ottenuti sia con l'algoritmo "standard" sia

con delle variazioni di tali algoritmo sviluppate nell'ambito della tesi tendento conto della stocasticità intrinseca insita nella valutazione della prestazione di una soluzione.

Più precisamente analizzeremo la possibilità di variare il numero di epoche usate per stimare la prestazione del sistema durante il processo di apprendimento allo scopo di verificare se una diminuzione del numero di epoche durante la fase iniziale del processo di adattamento possa essere utilizzato per introdurre stocasticità nel meccanismo di selezione necessaria per aumentare il fattore di esplorazione. Inoltre analizzeremo se un aumento del numero delle epoche durante la fase finale del processo di adattamento possa essere utilizzato per ridurre la stocasticità nel meccanismo di selezione in modo da ottimizzare efficacemente la soluzione trovata nella prima fase. Analizzeremo inoltre l'efficacia di un meccanismo di re-test della prestazione della soluzione corrente che possa ridurre i problemi causati dalla sovrastima della prestazione della soluzione corrente.

Esperimento 1

In un primo set di esperimenti abbiamo applicato l'algoritmo standard al setup sperimentale descritto nel capitolo 1. In ciascun esperimento, i parametri liberi del sistema vengono generati casualmente all'inizio del processo generando la soluzione iniziale del problema. Tale soluzione viene poi variata per 500 volte, scartando le variazioni non adattive e ritenendo le variazioni adattive. Ciascuna soluzione variata viene generata rimpiazzando il 2% dei bit della soluzione corrente con dei valori generati casualmente. Le modifiche introdotte in ogni variazione della soluzione vengono ritenute quando queste portano ad un miglioramento della fitness oppure, qualora fossero delle variazioni maladattive, con una probabilità che è funzione della temperatura e della differenza tra la fitness corrente e quella della soluzione variata. La temperatura parte da un valore iniziale T (pari a 0.005 e 0.02) e descresce linearmente fino a 0 durante il processo di adattamento che consta di 500 cicli. Per ciascun esperimento sono state effettuate 50 repliche partendo da soluzioni iniziali casualmente diverse.

La tabella 3.1 e le figure 3.2 e 3.3 mostrano i risultato ottenuti in 4 set di esperimenti in cui la temperatura iniziale è stata fissata a 0.02 e 0.005, rispettivamente, e in cui il numero di epoche utilizzate per stimare la prestazione della soluzione corrispondente è stato fissato a 20 e 60, rispettivamente.

	T = 0.02	T = 0.005
20 epoche	1.3872	1.4404
60 epoche	1.3420	1.3204

Tabella 3.1: Prestazioni massime ottenute variando il numero di epoche e il valore iniziale della temperature. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 200 epoche.

I risultati ottenuti confermano la relazione additiva tra la stocasticità introdotta attraverso la probabilità di accettare variazioni non adattive regolata dalla temperatura e la stocasticità dovuta alla inaccuratezza della stima della prestazione. In particolare si può osservare come, negli esperimenti in cui il

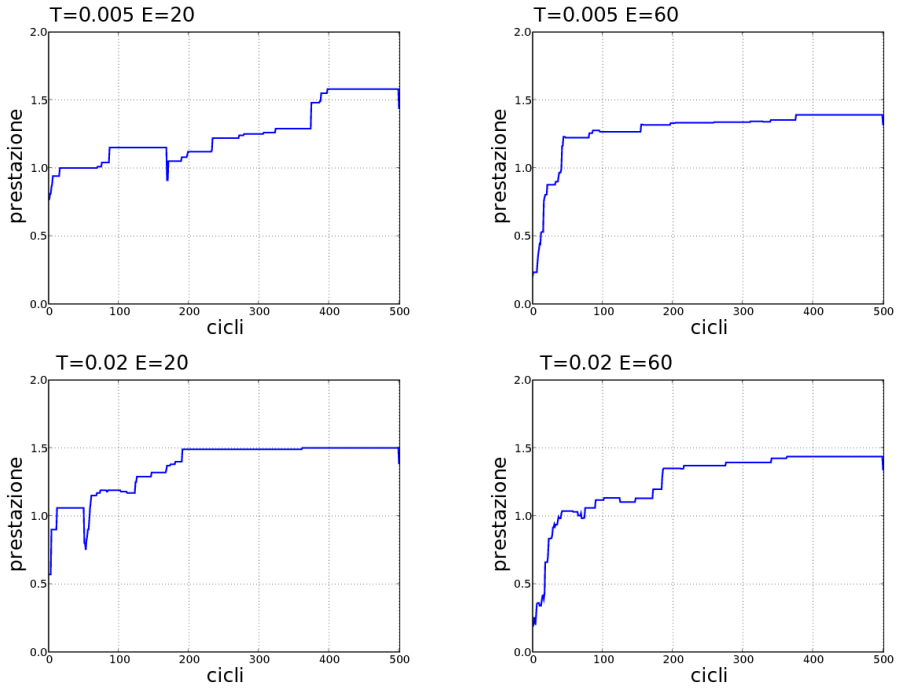


Figura 3.2: Prestazioni durante il processo di adattamento delle repliche migliori nelle quattro diverse condizioni sperimentali.

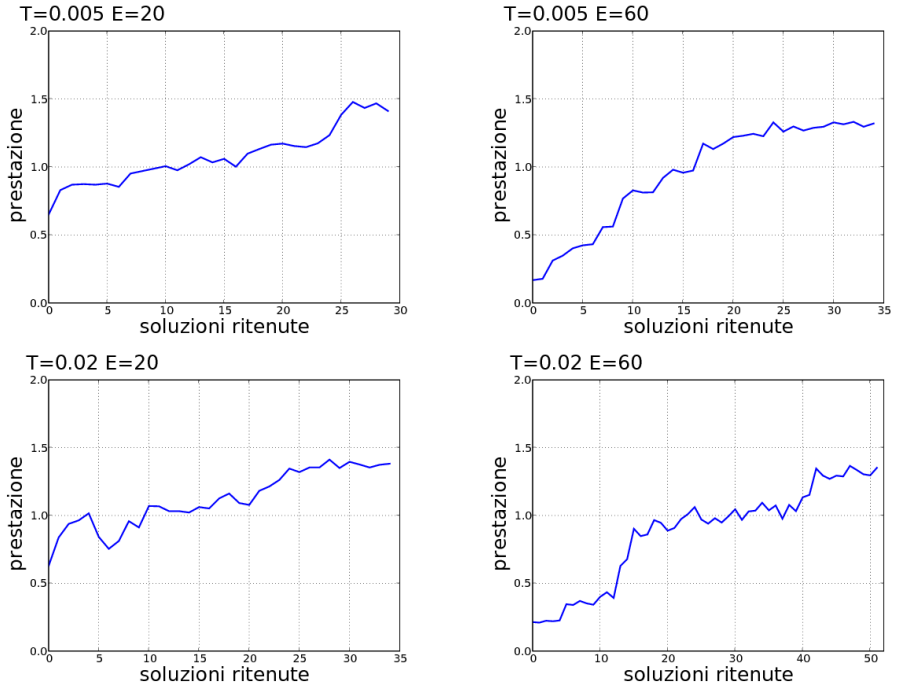


Figura 3.3: Prestazioni durante il processo di adattamento delle repliche migliori nelle quattro diverse condizioni sperimentali. I valori visualizzati si riferiscono alla stima ottenuta ritestando la prestazione delle soluzioni ritenute (cioè esclusivamente le soluzioni in cui le variazioni casuali introdotte sono state ritenute) per 200 epoche.

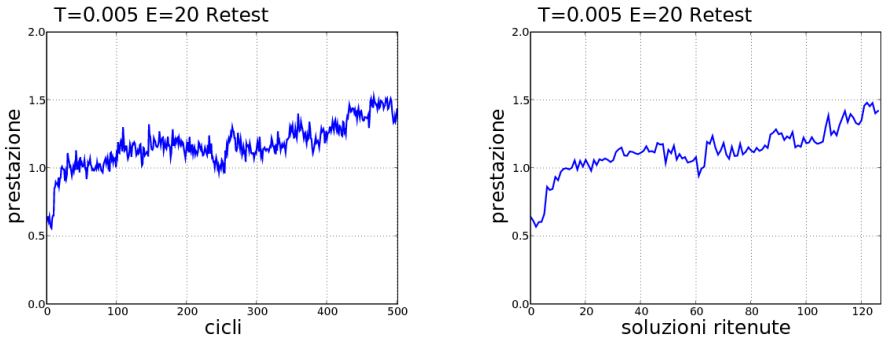


Figura 3.4: Prestazioni durante il processo di adattamento delle repliche migliori negli esperimenti eseguiti con 20 epoche e con il retest. I valori visualizzati si riferiscono alla stima ottenuta ritestando la prestazione delle soluzioni ritenute (cioè esclusivamente le soluzioni in cui le variazioni casuali introdotte sono state ritenute) per 200 epoche.

numero di epoche è relativamente basso (20) e dunque l'errore sulla stima della prestazione è alto, i risultati migliori si ottengono con una temperatura bassa (0.005). Al contrario, negli esperimenti in cui il numero delle epoche è alto (60) e dunque l'errore sulla stima della prestazione è più elevato, i risultati migliori si ottengono con una temperatura alta (0.02). Il fatto che i risultati migliori in assoluto si ottengono nel caso degli esperimenti con 20 epoche, mostra come l'aumento del numero di epoche (e di conseguenza del costo computazionale) non sia necessariamente un vantaggio e possa addirittura rivelarsi svantaggioso quando la stocasticità complessiva presente nel processo di selezione delle variazioni è troppo bassa.

Si può notare come alcune mutazioni che appaiono in figura 3.2 come modifiche adattive (che corrispondono quindi nel grafico ad una crescita nella fitness) appaiano come maladattive in figura 3.3 (una decrescita nella fitness), dove la stima è più accurata. Questo mostra l'effetto della stocasticità insita nel sistema in atto portando ad una sovrastima della prestazione.

Esperimento 2

Successivamente, abbiamo implementato e testato una variazione dell'algoritmo standard in cui la prestazione della soluzione corrente viene ritestata (per lo stesso numero di epoche) ogni volta che una soluzione variata viene scartata. La prestazione della soluzione corrente viene conseguentemente rivalutata mediando la stima ottenuta precedentemente con la stima corrente. Questo processo, che porta ad una riduzione dell'errore sulla stima della prestazione della soluzione corrente (ogni volta che soluzione corrente risulta essere migliore della solzione variata) serve ad impedire che una sovrastima della soluzione corrente impedisca l'accettazione di variazioni adattive riducendo o addirittura bloccando il processo di esplorazione.

Come si può vedere dalla figura 3.4, i risultati in termini di prestazioni sono simili con e senza il meccanismo del retest e confermano il dato che quando il numero di epoche e' basso (20) si ottengono risultati migliori con una tempera-

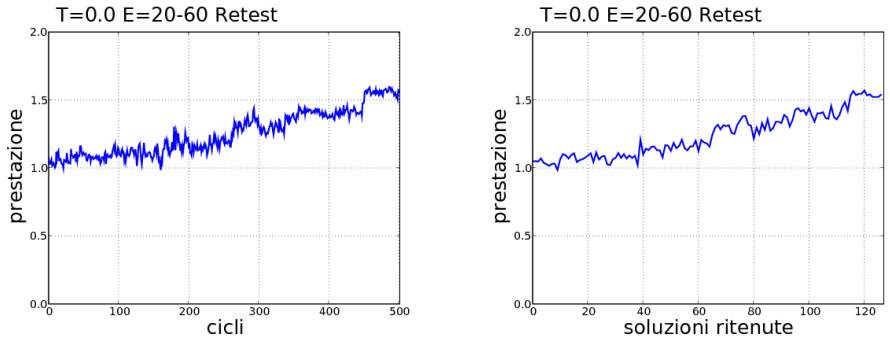


Figura 3.5: Prestazioni della replica migliore dell'esperimento in cui il numero di epoche varia da 20 a 60 durante il processo di adattamento e in cui la prestazione della soluzione corrente viene ritestata ogni volta che una soluzione variata viene scartata. La figura di sinistra mostra le stime della prestazione misurate durante il processo di adattamento in base al numero di epoche corrente. La figura di destra mostra le prestazioni di tutte le soluzioni ritenute testate per 200 epoche.

tura più bassa (0.005). Tuttavia, come si può vedere confrontando le figure 3.3 e 3.4, il meccanismo di retest produce un aumento significativo della capacità di esplorazione in quanto il numero di variazioni accettate è circa il quadruplo rispetto agli esperimenti senza retest.

Esperimento 3

Infine abbiamo implementato e testato una ulteriore variante dell'algoritmo in cui il meccanismo che determina la probabilità di accettare variazioni non adattive (in base al valore corrente della temperatura) è stato rimosso e rimpiazzato da una variazione del numero di epoche durante il processo di adattamento. In particolare abbiamo eseguito un esperimento in cui il numero di epoche varia linearmente da 20 a 60. Così come nel caso dell'esperimento precedente, le soluzioni ritenute vengano ritestate ogni volta che una soluzione variata viene scartata.

La figura 3.5 mostra come i risultati ottenuti con questa variante dell'algoritmo sono migliori di tutti quelli ottenuti precedentemente. Tale risultato supporta l'ipotesi che il modo migliore per consentire l'esplorazione di soluzioni diverse durante la prima fase del processo di adattamento e di ottimizzare la soluzione trovata nella seconda fase del processo di adattamento consista nel regolare la stocasticità insita nella stima della prestazione agendo sul parametro che controlla il periodo nel quale viene stimata la prestazione e di conseguenza l'accuratezza della stima. Il risultato inoltre supporta l'ipotesi che il retest della soluzione corrente sia necessario per evitare gli effetti di una sovrastima della prestazione della soluzione corrente.

3.2.2 Analisi quantitativa della stocasticità intrinseca nella stima della prestazione

Per analizzare quantitativamente il valore della stocasticità intrinseca al processo di stima della prestazione e la dipendenza di tale stocasticità dal numero

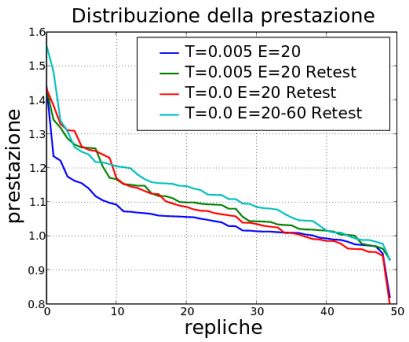


Figura 3.6: Distribuzione delle prestazioni stimate su 100 epoche alla fine del processo di adattamento. Ciascuna curva indica la distribuzione delle prestazioni di 50 repliche per ciascuna condizione sperimentale ordinate in ordine decrescente.

delle epoche abbiamo calcolato la probabilità di accettare variazioni neutre o maladattive in presenza e in assenza di una temperatura.

Per mostrare il parallelismo tra i due metodi abbiamo preso un individuo che avesse appreso il comportamento con una fitness di 1.58 su 500 cicli di SA ed abbiamo eseguito 1000 valutazioni per ognuno di essi.

Per mostrare l'effetto della variazione linearmente crescente del numero di epoche abbiamo costruito una matrice $D(500 \times 1000)$, per ogni riga i-esima della matrice D (ogni i-esima riga contiene le 1000 stime effettuate su tale soluzione) abbiamo estratto casualmente 1000 campioni $x = (x_1, x_2, ..., x_e)$ di dimensione e (che varia linearmente $f_e(c) = e_{min} + \frac{c \cdot (e_{max} - e_{min})}{N}$) e ne abbiamo calcolato la media ed inserite in una matrice $M(500 \times 1000)$. Ogni elemento della i-esima riga di M rappresenta una stima della prestazione dell i-esima soluzione fatta su e valutazioni.

Per ogni riga i di M abbiamo calcolato la media campionaria e la varianza campionaria per mostrare come la deviazione standard diminuisca al crescere del numero di epoche, e quindi della dimensione dei campioni (come indicato dalla legge debole dei grandi numeri).

In linea teorica, utilizzando la disequazione di Chebyshev:

$$P(\mu - \lambda\sigma \leq X \leq \mu + \lambda\sigma) \geq \ 1 - \frac{1}{\lambda^2} \tag{3.1}$$

(equivalente a $P\left(|x_i - \mu| \geq \lambda \cdot \sigma\right) \leq \ \frac{1}{\lambda^2}$)

si può calcolare la probabilità di accettare una mutazione che sia $\delta\%$ minore di quella precisa (su 1000 valutazioni) come:

$$\lambda\sigma = \delta \cdot fit \tag{3.2}$$

$$P(newfit - fit \geq \delta \cdot fit) \leq (\frac{\sigma}{\delta \cdot fit})^2 \cdot \frac{1}{2} \tag{3.3}$$

La media campionaria fit e la deviazione standard sono estratte da M. Questa probabilità decresce al decrescere di σ e al crescere di $\delta \cdot fit$. Allo stesso modo, la probabilità di accettare una mutazione negativa nell algoritmo

standard decresce al decrescere di T ed al crescere della differenza tra le fitness (nuovamente $\delta \cdot fit$).

Per la natura simmetrica del rumore esiste la stessa probabilità di non accettare una soluzione adattiva che per via dell'aprossimazione risulti essere maladattiva.

Nel caso dell'algoritmo standard in sistemi embodied and situated la stocasticità deriva da due fonti additive. Esiste la probabilità $P_{temp}(c)$ derivante dal criterio legato a T precedentemente descritto più la probabilità $P_{stoc}(c)$ di accettare una soluzione maladattiva dovuta all'errore di aprossimazione appena descritto. Nel caso dell algoritmo standard tuttavia la dimensione del campione non sarà variabile ma costante ed uguale al numero di epoche raggiunto alla fine del processo dall algoritmo modificato ($e_{max} = 60$). La probabilità totale di accettare una mutazione non adattiva nel algoritmo standard in sistemi embodied and situated sarà quindi:

$$P_{tot}(c) = P_{temp}(c) \cdot (1 - P_{stoc}(c)) + P_{stoc}(c) \tag{3.4}$$

Intuitivamente essa equivale alla probabilità di accettare la soluzione maladattiva per via dell'errore di stima più la probabilità di accettarla per via del criterio esplicito dato che tale soluzione non è stata sovrastimata.

Come precedentemente, data la natura simmetrica del rumore, esiste la probabilità di non accettare soluzioni adattive che risultino maladattive per via dell'errore dovuto all'aprossimazione della stima. Questa volta tale soluzione tuttavia ha ancora la possibilità di essere accettata per via del criterio dato dalla temperatura:

$$P_{act}(c) = P_{stoc}(c) \cdot P_{temp}(c) \tag{3.5}$$

Nel grafico 3.7 abbiamo calcolato, secondo la formula standard, la probabilità di accettare una soluzione minore del 2% rispetto a quella corrente (calcolata, su un campione molto grande di 1000 valutazioni, con $exp(\frac{(fit-0.98 \cdot fit)}{T})$). A questa probabilità abbiamo aggiunto, seguendo 3.4, la probabilità stocastica data da un campione di dimensione 60. Come si può notare, la probabilità è molto alta inizialmente e decresce fino ad esaurirsi completamente nell'ultima parte dell'apprendimento, andamento tipico per il SA. In figura 3.7 viene mostrato il grafico della stima dell'individuo con la deviazione standard con $e = 20$ e si può notare come il range di stime ottenibili da una soluzione sia spesso più ampio della variazione ottenuta accettando una variazione. Inoltre in figura 3.7 si può notare l'andamento molto simile della stocasticità nei due algoritmi.

Questo conferma la nostra ipotesi che prevede di poter sostituire il criterio esplicito di accettazione nell algoritmo standard con la variazione del numero di epoche con un vantaggio computazionale consistente.

Per poter utilizzare il simulated annealing standard in sistemi situated e emobodied è richiesto allo sperimentatore di utilizzare per tutto il processo un numero abbastanza elevato di epoche da minimizzare la stocasticità intrinseca al sistema, di modo che il fattore additivo non vanifichi la strategia.

Al contrario, utilizzando un numero variabile di epoche, solo alla fine del processo è necessario raggiungere una stocasticità nulla (con un numero di epoche finale uguale quindi a quello che verrebbe utilizzato con l'algoritmo standard durante tutto il processo).

La complessità del SA standard nella versione embodied and situated è proporzionale alla lunghezza del processo n ed al numero di valutazioni effettuate

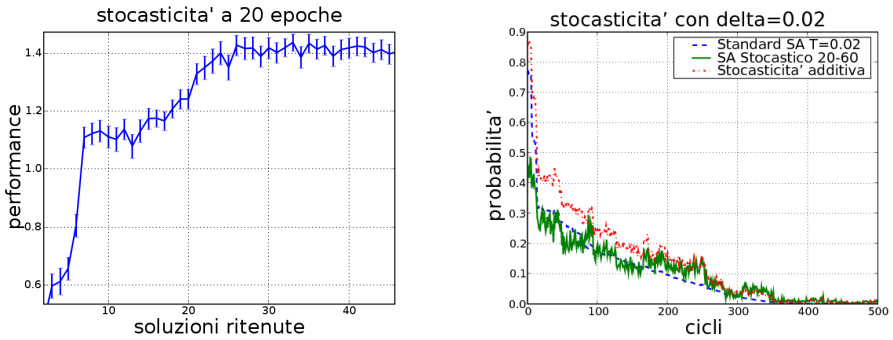


Figura 3.7: Stocasticità a 20 epoche. La barra di errore mostra il range di valori (la deviazione standard) che la stima della fitness può assumere utilizzando 20 epoche. Andamento della probabilità di accettare una stima minore del 2% con il SA standard e quello modificato.

per ciclo m : $O(n \cdot m)$. Si consideri che in tale algoritmo la componente più onerosa in termini di tempo computazionale è la valutazione. Per questo motivo trascuriamo i costi costanti come l'operatore di mutazione.

Mentre nell'algoritmo standard m è costante ed equivale ad e_{max}, nella versione ad epoche variabili essa varia linearmente tra e_{min} ed e_{max}. La somma totale delle valutazioni è quindi:

$$\sum_{i=1}^{N}(e_{min} + \frac{i \cdot (e_{max} - e_{min})}{N}) \tag{3.6}$$

Consideradando $e_{min} = \alpha e_{max}$

$$\sum_{i=1}^{N}(\alpha e_{max} + \frac{i \cdot (1-\alpha) e_{max}}{N}) \tag{3.7}$$

$$\sum_{i=1}^{N}(\frac{e_{max} \cdot (N\alpha + i \cdot (1-\alpha))}{N}) \rightarrow e_{max}(N\alpha + (1-\alpha)\frac{(N-1)}{2}) \tag{3.8}$$

La complessità dell'algoritmo standard (con $m = e_{max}$) è quindi $O(n \cdot e_{max})$, quella dell algoritmo modificato è:

$$O(e_{max}(n\alpha + (1-\alpha)\frac{n-1}{2})) \tag{3.9}$$

Con $\alpha = 1$ la complessità è $O(n \cdot e_{max})$ mentre con $\alpha = 0$ essa è $O(\frac{n \cdot e_{max}}{2})$.

La differenza computazionale dipende quindi dal rapporto α tra il numero minimo e quello massimo di epoche.

Non esiste un criterio preciso per scegliere α. Tuttavia si tenga conto che è esso a determinare la quantità di stocasticità iniziale e deve essere quindi sensibilmente minore di $\frac{1}{2}$.

Con $\alpha = 0$ il vantaggio computazionale è del 50% mentre nel caso degli esperimenti eseguiti in questo lavoro è stato utilizzato $\alpha = \frac{1}{3}$ con un vantaggio computazionale del 33%.

Il seguente pseudo-codice descrive l'algoritmo modificato:

```
c := 0
curr_perf := 0
curr_sol  := random_solution()
while c < learning_cycles:
    epochs := min_epochs + (c*(max_epochs-min_epochs))/learninc_cycles
    new_sol  := mutate(curr_sol)
    new_perf := test(new_sol)
    if new_perf >= curr_perf:
        curr_sol  := new_sol
        curr_perf := new_perf
    else:
        curr_perf := (curr_perf + test(curr_sol)) / 2
    c := c + 1
```

3.2.3 Sommario

In questa sezione abbiamo dimostrato come il SA possa essere utilizzato con successo nel setup sperimentale studiato in questa tesi. Inoltre abbiamo introdotto una variante dell'algoritmo standard che si è dimostrata siìgnificativamente più efficace dell'algoritmo originario.

3.3 Algoritmi evolutivi

Gli algoritmi evolutivi sono delle tecniche di ottimizzazione ispirate all'evoluzione naturale darwiniana che nel tempo ha portato a molte applicazioni in svariati campi[3].

Tipicamente un problema di ottimizzazione consiste nella ricerca di una configurazione $\vec{x} = (x_1, ..., x_n) \in M$ di parametri liberi appartenenti al sistema in considerazione che massimizzi (o minimizzi) un criterio $f : M \rightarrow R$ (detta funzione obiettivo o funzione di fitness o funzione di valutazione) $f(\vec{x}) \rightarrow max$.

Generalmente gli algoritmi genetici si basano su una popolazione inziale di soluzioni che viene soggetta a degli operatori probabilistici come la selezione, la mutazione e (ogni tanto) la ricombinazione (anche detta riproduzione o cross-over) per evolvere incrementalmente gli individui verso valori di fitness maggiori. La fitness di un individuo rispecchia il suo valore nei termini della funzione obiettivo da massimizzare[20, 19].

Ogni individuo della popolazione rappresenta un punto nello spazio dei parametri. La mutazione sposta l'individuo nel suo intorno in una direzione e di una distanza entrambi casuali in un range definito dal tasso di mutazione adottato introducendo innovazione, mentre la ricombinazione attua generalmente uno scambio di informazione tra gli individui. La selezione opera sui valori di fitness di ogni individuo e fa sì che l'algoritmo esplori maggiormente le zone più promettenti dello spazio facendo riprodurre gli individui con fitness migliori. Tuttavia la funzione di selezione deve garantire la presenza di diversità all'interno della popolazione per evitare che la ricerca rimanga bloccata in minimi locali.

Gli aspetti piu' importanti che caratterizzano diversi famiglie di algorimi evoluti sono: gli operatori di variazione e il metodo di selezione. Per quanto riguarda gli operatori di variazioni, in alcuni casi viene utilizzato solo un operatore di mutazione. In altri casi anche operatori come il crossing over. Per quel che riguarda i meccanismi di selezione, in alcuni casi si utilizzano meccanismi deterministici (come la truncation selection) mentre in altri casi si utilizzano meccanismi stocastici come la roulette wheels selection. Cosi' come nel caso del SA standard, la stocasticità viene introdotto al fine di accettare in alcuni casi anche variazioni maladattive che possono consentire di uscire da minimi locali. Un altro aspetto che influisce sulla stocasticita' del meccanismo di selezione è costituto dall'uso o meno dell'elitismo (cioè della conservazione delle soluzioni invariate senza variazioni nella generazione successiva). La mancanza di elitismo infatti introduce un ulteriore elemento di stocasticità in quanto non permette di mantenere la soluzione migliore invariata. Normalmente il tasso di variazione e' costante. Tuttavia nel caso delle evolutionary strategies il tasso di mutazione varia secondo un criterio statistico.

L'evolutionary strategies è una tecnica di ottimizzazione che consiste in un ciclo iterato di mutazione, che nel caso di spazi a valori reali avviene attraverso l'addizione di un valore estratto da una distribuzione normale, e selezione che è deterministico e prevede l'accettazione sul ranking della fitness. Una peculiarità di questa classe di algoritmi è il carattere adattivo del tasso di mutazione. Esso infatti varia durante la fase di apprendimento a seconda dell'andamento del processo (utilizzando ad esempio la matrice di covarianza dei parametri per facilitare le scelta dei parametri da variare).

Generalmente dal padre vengono generati tramite mutazione λ figli; nel caso

di $(1 + \lambda)$-ES i figli competeno con il padre per la paternità della generazione successiva, mentre nel caso di $(1, \lambda)$-ES il miglior mutante diventa padre della generazione successiva ed il padre precedente viene scartato. Esistono versioni che prevedono μ padri ed un operatore di ricombinazione (spesso chiamati $(\mu/\rho+, \lambda)$-ES) che dovrebbero essere più efficaci contro i minimi locali.

In [13] viene utilizzato un algoritmo ibrido in due fasi per la calibrazione di un sistema stereo vide camera-based. L'approccio utilizzato, chiamato *EES*, prevede una prima fase in cui la zona di ricerca viene selezionata, attraverso un $(1 + 1)$-ES, e successivamente esplorata attraverso $(1 + \lambda)$-ES con adattamento tramite matrice di covarianza.

Negli algoritmi evolutivi molteplici fattori favoriscono la stocasticità: lo schema di selezione, la presenza del cross-over, il tasso di mutazione, la dimensione della popolazione.

Il cross-over non è sempre utile in quanto prevede che vi sia una certa mappatura tra la rappresentazione nello spazio dei parametri e quello del comportamento. La mutazione permette l'esplorazione dell'intorno di una soluzione (a meno di tassi elevati) mentre la dimensione della popolazione aiuta l'estensione della ricerca ed agevola l'emergenza di una soluzione (con a monte una buona distribuzione dei genotipi di partenza). Lo schema di selezione è un elemento cruciale: schemi differenti applicano diverse pressioni selettive ed una diversa perdita di diversità nella popolazione spostando l'ago della bilancia del exploration/exploitation. I punti discussi nel primo capitolo riguardo le problematiche legate a sistemi di agenti embodied e situated si applicano anche agli algoritmi evolutivi. Gli algorimi evolutivi sono stati applicati virtualmente ad ogni possibile area dell'ottimizzazione inclusa l'evoluzione di sistemi embodied e situated. Questa applicazione rappresenta in realtà una vero e proprio campo di ricerca noto con il nome di Robotica Evolutiva[29].

A differenza del campo dell'ottimizzazione computazione dove sono state introdotte molteplici strategie di selezione per introdurre stocasticità che guidi il processo fuori dai minimi locali, nel campo dell'evolutionary robotics, o più in generale nei sistemi di agenti embodied e situated, poca enfasi viene messa sul tipo di schema utilizzato non considerando quindi il ruolo della stocasticità intrinseca a questi tipi di sistema.

Nell'ambito della robotica evolutiva l'approccio più diffuso è congruente con quello seguito in questa tesi. Vale a dire che l'algoritmo evolutivo viene utilizzato per addestrare il sistema di controllo di un robot o di un gruppo di robot. Tale sistema di controllo è costituito normalmente da una rete neurale artificiale. Dal punto di vista del meccanismo di selezione si utilizzano sia metodi stocastici come la roulette wheel selection sia metodi non stocastici (come il truncation selection). Analogamente si utilizzano metodi di selezione con o senza elitismo. Il problema della stocasticità della stima della prestazione viene risolto implicitamente stimando la prestazione su un numero considerevole di epoche e variando casualmente la posizione iniziale del robot e/o le caratteristiche dell'ambiente. Paradossalmente, nonostante l'ampiezza di questo campo di ricerca, all'autore non risulta tuttavia che il problema della stocasticità insita nella stima della prestazione sia stato studiato in modo sistematico. E in effetti, da questo punto di vista, l'utilizzo di meccanismi di selezione stocastici, appare immotivato.

Behavior-based robotics si basa sull'idea di provvedere ad un robot di alcuni comportamenti di base facendo in modo che il comportamento globale sia

il risultato dell'interazione di quelli sottostanti con l'ambiente [10, 12, 4]. Allo stesso modo evolutioary robotics pone un'enfasi sull'aspetto comportamentale del robot ma, per definire il comportamento globale, basandosi su un proceso automatico, solitamente fa un uso maggiore del trial & error ed a differenza del behavoir-based robotics la divisione in sottocomportamenti non avviene da parte dello sperimentatore ma è un processo auto-organizzante. Infatti l'organizzazione dell'intero sistema evolutivo, inclusa quinda la divisione dei sottoproblemi, è il risultato del processo di adattamento che normalmente prevede un grande numero di valutazioni di interazioni del sistema di controllo con l'ambiente[29, 25].

In ER due schemi di selezione vengono maggiormente utilizzati: il Truncation Selection e il Roulette-Wheel Selection. Il primo prevede che solo i migliori N individui possano riprodursi garantendo quindi la scelta dei migliori genotipi all'interno della popolazione. Il secondo prevede che gli individui vengano selezionati con una probabilità proporzionale alla loro fitness. Individui con fitness più elevata hanno maggiore probabilità di essere selezionati. Per questo motivo questo schema è anche spesso chiamato Proportional Selection.

3.3.1 Applicazione di un algoritmo evolutivo al setup sperimentale scelto

In questa sezione descriviamo un'applicazione di un algoritmo evolutivo standard al setup sperimentale scelto. Inoltre descriviamo i risultati ottenuto con una variazione di tale algoritmo che tiene conto del problema della stocasticità implicita nella stima della prestazione e dell'opportunità di variare il tasso di stocasticità durante il processo di adattamento al fine di ridurre il problema dei minimi locali mantendo al tempo stesso una capacità di ottimizzare la soluzione trovata durante la seconda fase del processo di adattamento.

Abbiamo scelto lo schema Truncation Selection perchè non introduce ulteriore stocasticità mantenendo il tasso di mutazione al 2% (valore utilizzato anche negli esperimenti sul Simulated Annealing), una durata di 100 generazioni ed una popolazione di 60 individui. L'ambiente, l'architettura della rete così come la funzione di fitness sono gli stessi utilizzati nella sezione precedente per il Simulated Annealing. Ogni esperimento è stato replicato 10 volte ed è stata selezionata ed analizzata la replica con il miglior valore di fitness finale.

Epoche	**Fitness**
20	1.618
40	1.538
60	1.614
20-60	1.694

Tabella 3.2: Prestazioni massime ottenute variando il numero di epoche e il valore iniziale della temperature. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 200 epoche.

I dati della tabella 3.2 mostrano che i miglior risultati a livello di fitness finale sono nuovamente raggiunti dalla versione con numero variabile di epoche.

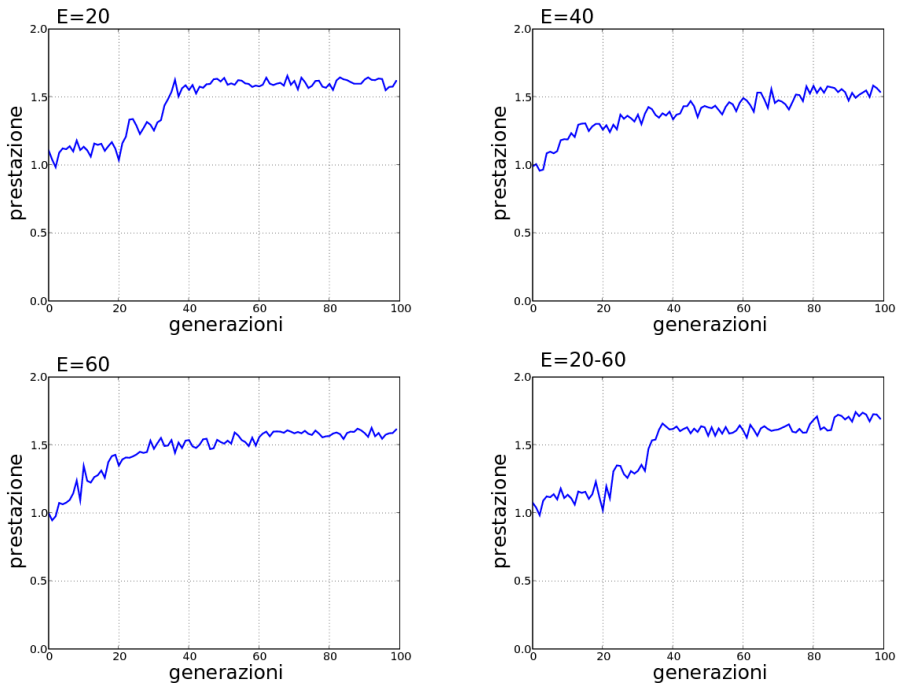


Figura 3.8: Andamento della miglior replica, di 10, dell'adattamento tramite algoritmo evolutivo con 4 valori di epoche diversi. Gli esperimenti sono stati eseguiti con stessi parametri ad eccezione del solo valore delle epoche.

Inoltre, analizzando l'andamento delle curve in figura 3.8, sembra di poter riconoscere una convergenza prematura nell'esperimento con minima stocasticità e sembra altrettanto riconoscibile la fase di alta esplorazione nella prima fase dell'esperimento a 20 epoche. I due esperimenti a stocasticità media (40 e 20-60) paiono entrambi essere ancora in crescita alla fine del processo di adattamento. I risultati sembrano mostrare che la variazione sul numero di epoche durante il processo possa fornire la stocasticità generalmente introdotta esplicitamente dalla funzione di selezione mantenendo un carattere più esplorativo nella prima fase e di affinamento nella seconda. Questi risultati confermano quanto concluso nelle sezioni precedenti mostrando vantaggi a livello di prestazione finale oltre che computazionali.

Inoltre i risultati ottenuti indicano come l'algoritmo proposto dall'autore possa rivelarsi più efficace degli algoritmi comunemente usati nell'ambito della robotica evolutiva.

3.4 Comparazione dei risultati ottenuti

L'opportunità di aver applicato un algoritmo di SA un algoritmo evolutivo allo stesso setup sperimentale ci da l'opportunità di fare alcune considerazione sulle prestazioni relative dei due algoritmi. Una comparazione dettagliata tra i due algoritmi rappresenta un'impresa non semplice che esula dagli obiettivi di questa tesi.

Per analizzare la differenza di costo computazionale tra l'apprendimento individuale e l'apprendimento evolutivo nel nostro caso situated e emobodied considereremo il simulated annealing come un algoritmo genetico che opera *online*, opera cioè con una popolazione con un solo individuo, senza crossover e senza selezione (se non il suo criterio di maggiore addattività) mantenendo quindi solo mutazione e valutazione.

Nel Simulated annealing l'esperimento eseguito è costituito da 50 repliche del processo di una durata di 500 cicli. Come specificato alla fine della sezione 3.2 l'elemento che varia all'avanzamento del processo è il numero di epoche che dipende dal valore minimo, quello massimo e dallo stato di avanzamento del processo. Considerando che il numero di epoche varia linearmente: $f_e(c) = e + \frac{c*(E-e)}{N}$, dove e ed E sono rispettivamente il numero minimo e massimo di epoche, e c (con $0 \leq c < N$) il numero di ciclo corrente, il numero di valutazioni che avvengono durante una replica è: $\sum_{i=0}^{N} f_e(i)$. Il numero di valutazioni che un individuo avrà nel corso di una relica sono quindi 19751 che moltiplicato per il numero di replice da 987550 valutazioni. L'operatore di mutazione, che ha costo costante, viene eseguito 25000 volte in 50 repliche.

Nel caso dell'algoritmo genetico abbiamo modificato i parametri dell'esperimento eseguito precedentemente nella sezione 3.3 per rendere simile il costo computazione (che con i precedenti parametri era del doppio rispetto a quello del SA) ed è costituito da 10 replice da 33 individui (11 padri che generano 3 figli l'uno) per 80 cicli. Nuovamente l'elemento variante è il numero di epoche che varia secondo gli stessi criteri. Ogni individuo ha una vita di 80 cicli quindi sarà valutato 3160 volte che moltiplicato per il numero di individui e di repliche da 1042800 valutazioni. L'operatore di mutazione e la funzione di selezione vengono eseguiti rispettivamente 26400 e 800 volte.

Il miglior individuo di questo esperimento ha raggiunto una fitness di 1.566 che paragonato alla fitness 1.560 ottenuta con in SA modificato mostra nessun vantaggio a livello di capacità di ricerca o di costo computazionale. Si consideri tuttavia che l'algoritmo evolutivo era anch'esso fornito di numero di epoche variabile nel tempo, a differenza dell'algoritmo standard, e presenta quindi il vantaggio di costo computazionale illustrato nella sezione 3.2.

Capitolo 4

Adattamento individuale e sociale

4.1 Introduzione

In questo capitolo analizzeremo la possibilità di sviluppare un algoritmo di apprendimento in grado di sfruttare l'interazione con altri individui soggetti ad apprendimento. In altre parole analizzeremo se la possibilità di apprendere in un contesto sociale possa comportare dei vantaggi dal punto di vista algoritmico. In particolare definiremo una funzione per descrivere le differenze comportamentali tra due agenti ed applicheremo l'algoritmo descritto nel capitolo precedente per minimizzare tale funzione obiettivo. Nella sezione 4.2 descriveremo parte della letteratura esistente sull'apprendimento sociale in natura, nella sezione 4.3, dopo aver introdotto la letteratura sull'apprendimento sociale in robotica, descriveremo l'algoritmo da noi proposto e gli esperimenti eseguiti. Infine nella sezione 4.4 analizzeremo i risultati ottenuti.

4.2 Apprendimento sociale

Per apprendimento sociale intendiamo quel processo attraverso il quale un individuo apprende una nuova capacità attraverso l'interazione con altri individui. In [5, 9, 17, 21] si può trovare la definizione e la discussione sulle varie forme di apprendimento sociale.

In natura, l'apprendimento sociale avviene a differenti livelli ed in diverse modalità[27]:

- **social facilitation**: un tipo di influenza sociale per cui un individuo esegue un comportamento quando in presenza di un conspecifico. Ad un osservatore esterno potrebbe sembrare che, dal momento che il comportamento viene esibito solo in presenza dei conspecifici, questo sia acquisito tramite imitazione dagli stessi. Invece la presenza dei conspecifici attiva tale comportamento con una regola come "non fare nulla a meno che non siano presenti alcuni conspecifici". Un possibile esempio sono alcuni animali che per mangiare attendono la presenza di conspecifici che potrebbero garantire un'attività di guardia.

- **contagious behavior**: per cui il comportamento di un altro è da stimolo per gli altri conspecifici ad eseguirlo, come il volo degli uccelli, l'abbaio dei cani etc. In questo caso l'esibizione del comportamento da parte dei conspecifici attiva lo stesso comportamento nell'individuo con una regola simile a "se gli altri abbaiano abbaia anche tu" etc. Questo tipo di comportamento sarebbe alla base tra le altre cose delle dinamiche di volo e di canto degli stormi di uccelli. Esso permetterebbe anche l'acquisizione di alcuni caratteri negli animali che l'utilizzano. Si pensi ad un uccello che abbia sviluppato la paura per un certo tipo di predatore e che voli via in sua presenza. Attraverso il contagious behavior ed una forma di apprendimento per associazione, come l'apprendimento Hebbiano, i conspecifici potrebbero apprendere tale tipo di reazione.

- **stimulus enhancement**: o local enhancement un fenomeno per cui stando in determinate situazioni ambientali si apprendono determinati concetti più facilmente per il semplice fatto di subire spesso un certo stimolo che altrimenti non verrebbe vissuto e risponde alla regola "segui qualcuno più vecchio di te ed impara da tutto ciò che accade". Si pensa sia attraverso tale tipo di apprendimento che alcuni ratti neri[38] apprendano come sbucciare le pigne per il fatto che stando vicino ai genitori essi vengano in contatto con pigne sbucciate solo parzialmente.

- **observational learning**: potrebbe rispondere alla regola "osserva da un individuo con esperienza il suo comportamento e se esso appare buono per lui, acquisisci quella strategia". Questo tipo di apprendimento sarebbe alla base dell'acquisizione delle paure in alcuni animali. L'osservazione della reazione alla paura di uno stimolo di un individuo esperto da parte di uno studente garantirebbe l'acqusizione di tale repulsione nello studente attraverso la sola osservazione. La valutazione del comportamento come buono o cattivo non sarebbe tuttavia necessario. Un esempio sarebbe l'acquisizione dei gusti sul cibo da parte dei ratti norvegesi che verrebbero trasmessi attraverso l'alito dei ratti esperti. Un giovane ratto imparerebbe quale cibo mangiare a seconda che ne abbia riconosciuto l'odore nell'alito dei ratti esperti attorno a lui. La regola sarebbe "osserva quale cibo viene mangiato dagli individui esperti e fai lo stesso".

- **matched-dependent behavior**: per cui un comportamento viene acquisito attraverso l'apprendimento con rinforzo ma la strategia acquisita veniva prima esibita da un individuo esperto. Questo non prevede quindi che lo studente sia consapevole delle intenzioni dell'individuo esperto. Si pensi ad un ambiente in cui dei ratti debbano apprendere tra due distributori di cibo quale contiene il cibo "buono". Il rinforzo verrebbe dato dal cibo stesso ma la scelta non sarebbe guidata dal trial & error ma dall'imitazione delle scelte di un individuo esperto. La regola potrebbe quindi essere "data una situazione percepita, esegui un comportamento che è risultato positivamente precedentemente".

- **cross-modal matching**: le forme di apprendimento descritte fino ad'ora non prevedono una traduzione tra due modalità sensoriali (due punti di vista) ma si basano su pattern sensoriali simili. Per comportamenti più complessi come l'acquisizione di una strategia che prevede l'utilizzo dei

> muscoli in un determineto modo risulta necessaria una capacità di traduzione tra il movimento dell'individuo esperto percepito e l'attivazione dei muscoli da parte dello studente. Ciò che viene percepito tuttavia non è di aiuto allo studente e prevederebbe un'analisi intenzionale da parte dello studente. Tuttavia il lavoro sui neuroni specchio[15] su primati ed umani ipotizzerebbe un qualche tipo di collegamento diretto tra il movimento percepito ed il movimento attuato permettendo quindi l'imitazione senza l'attribuzione di intenzioni da parte dello studente.

In termini generali l'apprendimento sociale viene solitamente definito nei termini della sua funzione di riduzione dell'incertezza[21]. L'apprendimento di una cultura, intesa in questo caso come un insieme di strategie, permette l'acquisizione di comportamenti adattivi in ambiene incerti, cioè in ambienti dove è costoso acquisire una strategia attraverso il trial & error[8]. Si pensi ad un ambiente dove apprendere lo stato dell'ambiente (ad esempio la bontà di alcuni cibi, la pericolosità di un predatore, la morfologia dell'ambiente scosceso circostante etc.) rappresenti un costo elevato nei termini della sopravvivenza per l'individuo. Quasi per definizione, l'adulto è un individuo che possiede conoscenze e strategie che gli permettono di sopravvivere all'interno del suo ambiente a differenza dell'individuo giovane che nasce con il solo bagaglio contenuto nel suo codice genetico. In questo caso acquisire da un altro individuo la categorizzazione del cibo o la repulsione verso certe zone dell'ambiente così come verso certi predatori permetterebbe ad un individuo studente di evitare i rischi (i costi) di acquisirli "sulla propria pelle" aumentando quindi la probabilità di sopravvivere. Tuttavia non sempre l'apprendimento sociale è vantaggioso rispetto all'apprendimento individuale. In ambienti instabili, cioè in ambienti dove le configurazioni dell'ambiente ed i costi di acquisizione dell'informazione variano nel tempo, risulta necessario acquisire lo stato dell'ambiente attraverso l'apprendimento individuale. Le strategie acquisite attraverso l'apprendimento sociale potrebbero infatti non essere più adattive per l'ambiente variato e delle decisioni basate su tali informazioni potrebbero rivelarsi controadattive[5, 9]. Per "riduzione dell'incertezza" si intende quindi l'acquisizione dello stato dell'ambiente circostante all'individuo (di informazione riguardo l'ambiente). In linea di principio l'utilizzo di apprendimento individuale, sociale ed evolutivo dipende dal rapporto tra la frequenza con cui varia l'ambiente (per quanto tempo l'informazione sullo stato dell'ambiente è valida) e la scala temporale con cui i tre tipi di apprendimento trovano una soluzione adattiva.

Solo attraverso l'apprendimento individuale un agente può adattarsi efficacemente ad un ambiente che varia più volte durante la sua vita, dal momento che la strategia acquisita socialmente all'inizio potrebbe risultare obsoleta. Allo stesso modo non è necessario che un agente apprenda invidivualmente quando le variazioni ambientali hanno una frequenza più bassa rispetto alla durata della vita dell'agente. L'apprendimento evolutivo è il più vantaggioso quando il tempo intercorso tra una variazione e l'altra equivale ad un grande numero di generazioni di agenti[21]. Questo tipo di considerazioni vale per gli ambienti in cui l'acquisizione di nuove informazioni rappresenta un costo per l'agente in termini di sopravvivenza. Un esempio di problema come quello appena descritto sarebbe l'acquisizione di una categorizzazione per i funghi velenosi.

4.3 Apprendimento sociale in sistemi embodied & situated

Un'applicazione dell'apprendimento sociale alla robotica è in linea con l'aumentare della complessità dei robots. Considerato l'ampio spazio parametrico che i robot presentano, volendo evitare gli approcci teleguidati o hard-coded, diventa difficile eseguire una ricerca iterativa tra i parametri che in un robot con 30 gradi di libertà (con 3 comandi per giunto, come *avanti, indietro e fermo*) raggiunge una combinazione di $3^{30} > 10^{14}$ azioni che possono essere eseguite in ogni stato. Questo è uno spazio state-action di ricerca troppo grande per cercavi una buona azione. Imparare un comportamento attraverso l'imitazione riduce drasticamente lo spazio di ricerca, perchè fornisce un ottimo punto di partenza per il processo di apprendimento dello studente[35]. Tali applicazioni tuttavia presuppongono dei sistemi non autonomi dipendenti dall'intervento dello sperimentatore umano. Nell'ambito della robotica e dell'Intelligenza Artificiale, l'apprendimento sociale è stato utilizzato allo scopo di sviluppare algoritmi che permettano ad un robot di acquisire delle capacità in interazione con un utente umano che "guida" il processo di apprendimento nel robot.

L'approccio behaviour-based, all'apprendimento tramite imitazione, si basa sulla generazione di N sottocomportamenti primitivi e chiusi (anche se controllabili parametricamente) da parte dello sperimentatore e da comporre a seconda del comportamento globale esibito dal modello che si vuole imitare. Ulteriori sottocomportamenti possono essere aggiunti, qualora essi non fossero disponibili ma richiesti dal goal finale. Per generare nuovi sottocomportamenti tra quelli inseriti dallo sperimentatore molteplici approcci sono possibili: l'approccio supervisionato permette allo studente di aprossimare la tabella *state/action* del modello, ma richiede informazioni sui suoi stati interni. Un altro approccio è basato sui sistemi competitivi di predizione [36, 26, 14, 41], approccio per cui multipli forward models (sistemi che dato uno stato del robot ne prevedono il successivo) competono per il controllo del robot a seconda dell'accuratezza della predizione che ogni modello fornisce. Questo approccio si basa sul concetto che un modello di predizione possa essere anche un buon modello di esecuzione del task, ispirandosi ai risultati sui lavori sui neuroni specchio[15]. Una volta generati questi sottocomportamenti, il robot deve comporli in una sequenza temporale che definirà il comportamento globale esibito. La ricomposizione dei sottocomportamenti avviene a seconda dell'approccio utilizzato. In quello simbolico vengono registrati i dati sensoriali ed i dati spaziali dei markers applicati al modello durante l'esecuzione dei movimenti del comportamento da imitare. Questi movimenti vengono divisi in segmenti di subgoals e di sottocomportamenti primitivi che permettono il loro raggiungimento. Il planner utilizzerà una forma di logica simbolica su questi segmenti per raggiungere il goal finale. Nell'approccio cinematico invece, i dati dei giunti e dei markers, nello spazio cartesiano tridimensionale, vengono utilizzati per calcolare le traiettorie da loro eseguite. A differenza dell'approccio simbolico, dove il planning avviene attraverso la logica simbolica, nell'approccio cinemantico vengono calcolati i parametri dei giunti che fanno eseguire al robot le traiettorie mostrate dal modello.[35]

In questa tesi invece, in linea con il lavoro di Acerbi e Nolfi[2, 1], ci si propone di sviluppare degli algoritmi completamente autonomi dall'intervento umano in cui i robot sviluppano le loro capacità autonomamemente e in cui le abilità

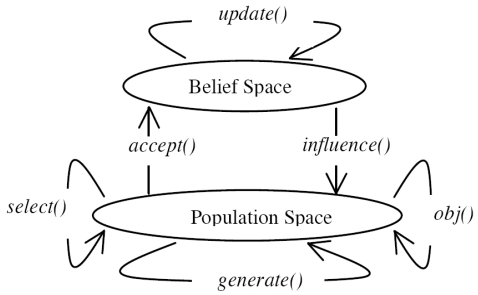


Figura 4.1: Framework per i Cultural Algorithms[33].

scoperte da alcuni robot individualmente possono favorire l'apprendimento di capacità analoghe in altri robot che non hanno ancora sviluppato tali capacità.

L'approccio utilizzato nell'algoritmo di ricerca è simile a quello usato dalla classe di algoritmi di ottimizzazione chiamata "Cultural Algorithms"[33]. Questi algoritmi presentano tre componenti principali: il population space, il blief space ed un protocollo che descrive il modo in cui l'informazione viene scambiata tra le prime due componenti. Il population space rappresenta lo spazio dei genotipi (dei parametri) degli individui e può supportare un qualunque algoritmo di adattamento basato su popolazione, come Genetic Algorithms o Evolutionary Programming. In figura 4.1 viene mostrato il framework per questo tipo di algoritmi. Il concetto su cui essi si basano è quello del "meme" [32], che costituisce una teorica unità di informazione culturale, il building block della cultura e dell'evoluzione culturale che si diffonde in modo analogo a quello con cui il gene si propaga da un organismo all altro come unità di informazione genetica. Questi algoritmi rappresentano quindi l'informazione culturale e l'informazione genetica all'interno dello stesso schema di rappresentazione: lo spazio dei parametri. Il belief space contiene informazioni sulla storia dei singoli parametri proveniente dai migliori individui di ogni generazione. Queste informazioni vengono utilizzate dall'operatore di mutazione che modifica il codice genetico degli agenti all'interno dei range specificati dal belief space, come aree dello spazio dei parametri che si pensa essere più promettenti. Tuttavia la differenza tra l'approccio appena descritto e quello utilizzato in questa tesi sta nel fatto che l'influenza del modello avviene direttamente sui parametri liberi del sistema. Come esporremo nella sezione 4.3.2, nell'approccio da noi proposto l'influenza del modello avviene indirettamente inquanto l'imitazione avviene su uno spazio differenti da quello dei parametri.

In [2] gli autori mostrano come una combinazione di apprendimento sociale ed individuale apporti vantaggi adattivi per agenti artificiali in sistemi embodied e situated che debbano sviluppare comportamenti troppo complessi o troppo costosi per essere sviluppati con apprendimento individuale. In modo analogo al lavoro svolto in questa tesi, gli agenti vengono lasciati vivere nell'ambiente simulato ed apprendere la categorizzazione di oggetti rappresentanti cibo "buono" o "cattivo" tramite un algoritmo simile al SA. Dopo una prima generazione di individui che apprendono individualmente viene selezionato l'individuo con la massima fitness, chiamato modello. La generazione successiva di individui, che partono con dei parametri casuali, apprende la strategia del modello attraver-

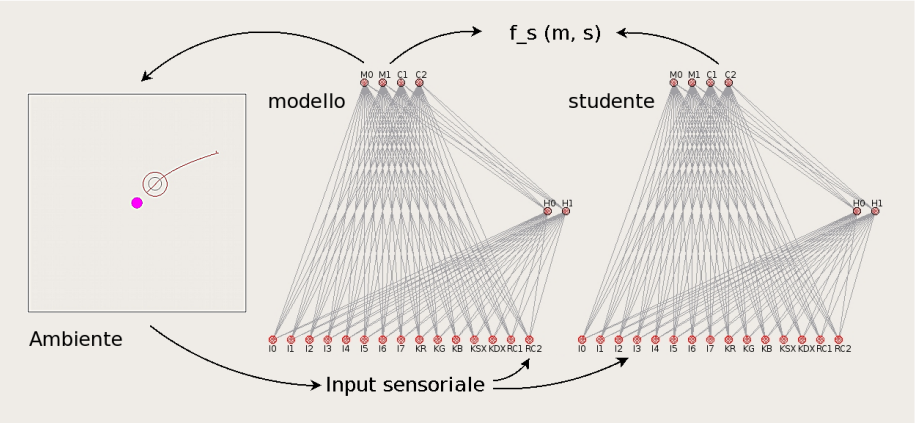


Figura 4.2: Il ciclo imitativo. Lo studente sta "sulle spalle del modello" ed entrambi si muovono secondo l'output del modello.

so l'algoritmo di apprendimento sociale. Nuovamente il miglior individuo della popolazione viene selezionanto diventando il nuovo modello per la generazione successiva ed il processo continua iterativamente. Gli autori mostrano come ad ogni generazione la strategia del modello venga gradualmente migliorata. Gli autori mostrano come la combinazione di una forma di apprendimento individuale in cui l'individuo cerca di migliorare le proprie prestazioni rispetto al compito dato con una orma di apprendimento sociale (in cui l'individuo cerca di acquisire un comportamento simile al modello) produca dei risultati migliori del solo apprendimento individuale.

Il setup sperimentale utilizzato dagli autori è tuttavia molto semplice. Infatti:

1. i robot sono posti in un ambiente toroidale privo di mura e ostacoli.

2. il numero di cibi da categorizzare è di soli 4 elementi di cui due buoni e due cattivi (descritti da una codifica booleana a 2 valori per oggetto).

3. i robot hanno a disposizione informazione sulla distanza dell'oggetto.

Inoltre occorre considerare che l'algoritmo di apprendimento utilizzato da Acerbi e Nolfi risulta chiaramente sotto-ottimale sulla base dei risultati descritti nel capitolo 3 in quanto si basa sul SA standard senza considerare i fattori della stocasticità intrinseca al sistema. Nell'ambito di questa tesi abbiamo dunque deciso di cercare di replicare questo tipo di esperimenti su un setup sperimentale piu' complesso al fine di valutare la scalabilità del modello utilizzando l'algoritmo di apprendimento ottimizzato elaborato e descritto nel capitolo precedente.

In questo capitolo (in modo analogo a quanto avviene in [2]) per apprendimento sociale intendiamo un algoritmo di apprendimento come quello presentato nel capitolo precedente, in cui la funzione da minimizzare sia la differenza sensomotoria tra un individuo che presenti già una soluzione adattiva (il modello) ed un individuo che presenti inzialmente una soluzione casuale (lo studente). Come precedentemente, l'algoritmo prevede la modifica dei parametri liberi del sistema di controllo dell'agente studente ed il mantenimento di tali modifiche a

patto che esse portino ad una maggiore somiglianza tra il modello e lo studente a livello di comportamento senso-motorio.

Per somiglianza a livello senso-motorio intendiamo la differenza tra gli output del modello e quelli dello studente, ai cui sistemi di controllo siano stati forniti gli stessi input sensoriali, in ogni istante di vita dell'agente:

$$f_s(m, s) = \frac{\sum_{i=1}^{U} |m_i - s_i|}{U}$$

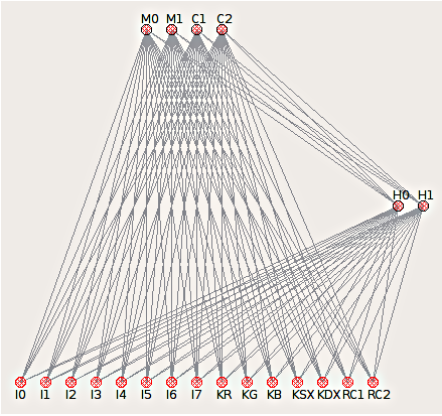
dove m_i, s_i sono l'i-esimo output del sistema di controllo rispettivamente del modello e dello studente ed U il numero di output presenti nel sistema di controllo.

4.3.1 Variazioni al setup sperimentale

Per gli esperimenti di adattamento sociale abbiamo eseguito delle modifiche al setup sperimentale (riassunte in figura 4.3):

- É stato complicato il problema inserendo 2 nuovi oggetti da categorizzare (1 buono e 1 cattivo).
- Sono stati aggiunti 2 neuroni aggiuntivi al output layer della rete neurale e due neuroni al input layer che ricevono l'attivazione precedente di questi due output aggiuntivi.

La prima modifica permette di complicare il problema. Dal momento che l'algoritmo individuale, come evidenziato nel capitolo precedente, riusciva a risolvere il problema con una categorizzazione corretta per più del 80% degli oggetti, abbiamo aggiunto due nuovi oggetti di modo da rendere lo spazio dei parametri più complicato da risolvere ed aumentare il rischio di cadere in minimi locali.



Buoni	Cattivi
61, 208, 208	27, 243, 17
15, 86, 174	11, 148, 146
254, 194, 62	234, 140, 234
208, 98, 194	187, 56, 89
223, 77, 98	247, 238, 45
170, 229, 10	110, 10, 253

Figura 4.3: L'architettura variata della rete neurale: l'output dei neuroni C1 e C2 al tempo t diventa l'input dei neuroni RC1 e RC2 rispettivamente, al tempo $t + 1$. La nuova codifica con l'aggiunta dei due nuovi oggetti (in fondo alle due colonne).

L'aggiunta dei due neuroni al output layer ed al input layer, con rispettiva connessione, è legata all'apprendibilità della strategia. Questi due neuroni aggiunti non hanno responsabilità motorie (non sono collegati a nessun tipo di attuatore) e non vengono contemplati dalla funzione di fitness. Essi nascono dall'idea che l'attivazione di questi due neuroni rispecchino l'attività della rete neurale e che quindi in qualche modo contengano informazione sulla categorizzazione effettuata dall'agente. Essendo il loro output collegato nuovamente a due neuroni di input, questi neuroni hanno per l'agente una funzione di memoria temporale che permette all'agente un comportamento più coerente nel tempo. Questi due neuroni aggiuntivi influenzano anche l'apprendimento sociale. Lo studente dovrà infatti imitare anche l'attivazione di questi due output aggiuntivi e questo, seguendo le considerazioni appena fatte, dovrebbe poterlo aiutare a creare una categorizzazione simile a quella del modello (in quanto accede indirettamente ad informazioni sullo stato interno del modello). Generare un comportamento simile per questi output aggiuntiv, considerando che esso sia legato alla categorizzazione interna al sistema di controllo, potrebbe aumentare la probabilità che anche lo studente generi una rappresentazione interna simile a quella del modello.

Dal momento che ad entrambi i sistemi di controllo vengono presentati, al tempo t, gli stessi vettori di input sensoriale che dipendono dall'ambiente, dalla posizione dell'agente e dall'azione intrapresa (l'output del sistema di controllo al momento $t-1$) dal sistema di controllo dell'agente modello, diremo che lo studente "sta sulle spalle del modello".

4.3.2 Applicazione dell'apprendimento sociale al setup sperimentale scelto

L'obiettivo del processo di apprendimento sociale è quello di consentire allo studente di acquisire rapidamente una strategia qualitativamente simile a quella del modello che può poi essere raffinata e migliorata attraverso l'apprendimento individuale. Per strategia intendiamo per es. che il robot studente acquisisca la capacità del modello di categorizzare gli stimoli visivi in due categorie distinte corrispondenti a due comportamenti distinti (di avvicinamento e di allontanamento) anche se i dettagli del comportamento del modello e dello studente variano. Per es. anche se il comportamento esibito dal modello e dallo studente per avvicinarsi o allontarnsi dall'oggetto non è identico.

Al contrario al sistema viene chiesto di trovare una soluzione indipendente, a livello parametrico, dalla soluzione utilizzata come modello e ciò lo spinge a creare, in modo emergente, la sue sottosoluzioni al problema in esame (il riconoscimento degli oggetti) in modo indiretto.

La generazione della soluzione al problema è solo un risultato della soluzione al problema imitativo. Minimizzando la differenza media tra i due sistemi di controllo si otterrà anche un sistema di controllo che sa risolvere il problema in esame.

Si noti come, anche nel caso in cui lo studente acquisisca la capacità di esibire lo stesso identiico comportamento del modello, i parametri liberi dei due robot possano essere significativamente diversi. La possibilità di trasmettere una capacità al livello del comportamento invece che a livello dei parametri liberi che producano il comportamento è potenzialmente interessante in quanto può consentire allo studente di partire da una soluzione diversa dal modello che potreb-

be risultare ulteriormente ottimizzabile attraverso il processo di apprendimento individuale.

Considerando un agente che debba apprendere un comportamento molto complesso l'apprendimento sociale permette all'agente di esplorare uno spazio parametrico ridotto evitando di spendere tempo computazionale in direzioni non promettenti o addirittura maladattive. L'apprendimento sociale può permettere quindi una velocizzazione nell'acquisizione di un comportamento rispetto all apprendimento puramente individuale. Oppure, volendo mantenere una durata costante (ed uguale costo computazionale), potrebbe permettere un innalzamento della fitness media di una popolazione di studenti che apprendono attraverso l'apprendimento sociale rispetto ad una popolazione che, a parità di condizioni, apprende individualmente.

Si noti come il segnale generato dalla funzione di somiglianza sia di natura differente rispetto al segnale generato dalla funzione di fitness. Accettare le variazioni sui parametri prendendo in considerazione questo segnale piuttosto che il segnale generato dalla funzione di fitness permette all'adattamento di accettare delle variazioni che sarebbero negative da un punto di vista della performance sul problema e che non verrebbero quindi accettate. Tuttavia queste modifiche permettono di visitare una parte dello spazio dei parametri (che verrebbe altrimenti esplorato con meno probabilità) promettente in quanto definirebbe un comportamento simile a quello del modello, che presenta una performance alta sul problema.

In qualche modo si può considerare quindi il sistema di controllo del modello come un modulo del sistema che genera un segnale, di natura emergente e quindi non progettato dall'alto, sulla bontà della strategia, ma non nei termini della performance sul problma.

Questo genere di approccio è in qualche modo analogo a quello utilizzato in [30] dove il sistema di controllo è fornito di un modulo interno che genera un segnale di rinforzo per il resto della rete. Questo modulo di rinforzo non è tuttavia fornito a priori di un'euristica per generare il segnale correttamente, ma il suo comportamento è anch'esso emergente e soggetto ad adattamento.

Come vedremo sucessivamente, questo aspetto è molto importante perchè permette di risparmiare tempo utile da impiegare nell'affinamento della strategia acquisita.

Nella parte restante di questa sezione esponiamo i tre esperimenti eseguiti in quest'ambito. Nel primo esperimento mostriamo come, attraverso il solo apprendimento sociale, l'agente non possa apprendere un comportamento adattivo al problema. Nel secondo esperimento mostriamo una versione modificata dell'algoritmo di apprendimento sociale al quale viene aggiunta una fase in cui l'individuo apprende sia individualmente che socialmente, risolvendo così la discrepanza emersa nell'esperimento precedente. Nel terzo esperimento mostriamo come l'apprendimento individuale sia più vantaggioso dell'apprendimento sociale, utilizzando l'attuale funzione di somiglianza, in termini di costo computazionale.

Esperimento 1: Apprendimento sociale

Abbiamo eseguito con il nuovo setup sperimentale un primo esperimento che prevede di far apprendere ad una popolazione di 50 individui la strategia sviluppata, attraverso l'apprendimento individuale, da un agente modello. In questo primo

esperimento gli individui sono stati sottoposti esclusivamente ad una forma di apprendimento sociale attraverso la quale cercano di imitare il comportamento esibito dal modello.

	Prestazione Modello	Prestazione Studente	Somiglianza
Migliore	1.5883	1.2066	0.9404
Media	1.2437	0.7017	0.9107

Tabella 4.1: Prestazioni del migliore individuo e prestazioni medie (calcolate su 50 individui) di individui che apprendono imitando il miglior individuo ottenuto attraverso una forma di apprendimento individuale. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 100 epoche.

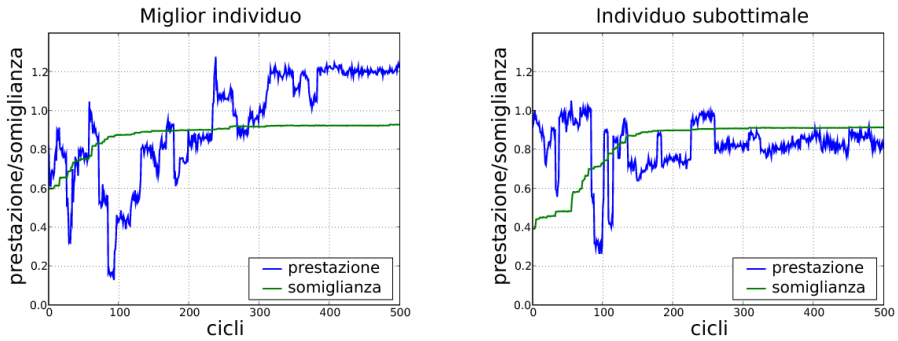


Figura 4.4: Individuo con fitness massima ed individuo estratto casualmente tra gli individui che apprendono socialmente. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 100 epoche.

A livello intuitivo ci si aspetterebbe che la curva di apprendimento individuale cresca al crescere della curva di somiglianza. Nella figura 4.4 tuttavia si può notare come l'andamento logaritmico della curva che rappresenta il grado di somiglianza tra lo studente ed il modello sia contrapposto ad un moto oscillatorio della curva sulla performance. Si noti che entrambi gli individui mostrati in figura raggiungono alla fine del processo un fattore di somiglianza di 0.92. Ciò indica che ad un miglioramento della somiglianza tra la strategia dello studente e quella del modello non corrisponde necessariamente un miglioramento della performance sul problema del quale la strategia dovrebbe essere una soluzione adattiva. Ciò mostra anche come un uguale fattore di somiglianza non indichi un uguale performance sul problema. La fitness media della popolazione alla fine del processo di apprendimento sociale si attesta su 0.7017 mentre la migliore su 1.2066, minori della fitness del modello (1.5883) e della fitness media della popolazione (1.2437) da cui il modello è estratto. Data la natura del sistema embodied e situated, bastano piccole differenze che riguardano le risposte motorie prodotte dal modello e dallo studente al livello delle singole interazione robot/ambiente possono portare a differenze considerevoli al lungo termine che si manifestano con valori di prestazione molto diversi. Molto dipende dalla

velocità utilizzata nell'esplorazione, l'angolo della traiettoria con cui ci si allontana dall ostacolo etc, e solo un affinamento della strategia sul tipo di problema (massimizzando la fitness) può risolvere queste imperfezioni sulla strategia.

Data la natura del processo imitativo e della sua fitness, le piccole differenze a livello senso-motorio possono portare in realtà a grandi differenze a livello di prestazioni. Si devono infatti tenere in considerazione i seguenti punti:

1. la fitness $f_s()$ descrive la differenza tra le due strategie senso-motorie. Questa è una differenza mediata sul numero di output, sul numero di epoche e prove (quindi sulla totalità degli oggetti da riconoscere) e sul numero di passi di cui una prova è costituita. Per questo l'impatto sulla performance che le differenze sensomotorie possono avere dipende molto dal contesto in cui esse emergono.

2. l'insieme di input sensoriali su cui la somiglianza viene calcolata è un sottoinsieme dei possibili input sensoriali ai quali l'agente può venire sottoposto nell'arco della sua vita all'interno dell'ambiente. I vettori di input dipendono dalla condizione iniziale e dalla strategia adottata dal modello.

3. la frequenza con cui i vari input sensoriali vengono presentati allo studente sono diversi e dipendono dalla strategia del modello.

Si pensi al problema di riconoscimento degli oggetti preso in esame in questa tesi. Si supponga ad esempio che la strategia appresa dal modello preveda una rotazione di 360° gradi seguita da una traiettoria lineare diretta che porta dal punto di partenza all'oggetto (categorizzato come buono) al centro dell'arena. Si supponga inoltre che la strategia eseguita con gli oggetti categorizzati come cattivi preveda un simile comportamento mantendosi però fermo una volta raggiunta la distanza minima di 150 unità. Con questo tipo di strategia lo studente verrà solo raramente sottoposto ad input provenienti dai sensori IR (su cui si basa la strategia di evitamento degli ostacoli) in prossimità dei muri. Per questo difficilmente il robot verrà valutato dalla funzione di somiglianza per la sua capacità di evitare le collisioni.

Inoltre, come specificato precedentemente, questo tipo di problema prevede di sviluppare una categorizzazione ed un buon comportamento motorio di approccio all oggetto. La funzione di somiglianza non distingue questi due comportamenti. Una buona categorizzazione esplicitata da una nuova strategia motoria (ad esempio un approccio tramite spirale piuttosto che lineare) verrà quindi considerata da questa funzione come una grande differenza di comportamento.

Per questi motivi non esiste una relazione diretta tra un alto fattore di somiglianza tra studente e modello ed una buona prestazione da parte dello studente. Si può quindi concludere che, sebbene l'apprendimento per imitazione possa consentire l'acquisizione di una strategia qualitativamente simile a quella del modello, difficilmente può consentire da solo l'aquisizione di una strategia efficace.

Si può quindi considerare, nei termini utilizzati nella sezione 2.2 riguardo gli algoritmi di ottimizzazione iterativa, l'apprendimento sociale come una tecnica di scelta delle configurazioni iniziali. Questa configurazione, per via della natura della soluzione trovata dall'apprendimento imitativo, dovrebbe trovarsi, nello spazio di ricerca di parametri, in un punto differente rispetto a quello rappresentato dal modello. Se il modello si fosse trovato in un minimo locale, questo

processo comporterebbe un salto fuori da questo stesso minimo, ma non sufficiente a determinare una soluzione altrettanto buona in termini di performance sul problema di categorizzazione. Questo meccanismo di salto dai minimi locali deriva dal fatto che lo studente parte da dei parametri casuali e non subisce una pressione ad assomigliare al modello da un punto di vista dei parametri ma è sottoposto ad una generazione guidata dei suoi parametri attraverso una comparazione delle differenze esibite dai due agenti a livello comportamentale.

L'algoritmo di apprendimento individuale può sfruttare la protostrategia generata dal processo imitativo ed affinarla per quell'insieme di input sensoriali al quale lo studente è stato raramente sottosposto, attraverso un processo di trial & error.

Esperimento 2: Apprendimento sociale e individuale

Per risolvere i problemi legati alla discrepanza tra la fitness ottenuta sul problema e la somiglianza con il modello abbiamo introdotto una seconda fase di apprendimento durante la quale l'agente viene valutato sia sul problema di somiglianza che sul problema di categorizzazione del cibo. Abbiamo diviso la unica fase di apprendimento della durata di 500 cicli in due fasi. La prima fase è costituita da 150 cicli di apprendimento sociale, come descritta nella sezione 4.2 mentre la seconda fase, di 350 cicli, è costituita da una miscela di apprendimento sociale ed apprendimento individuale. Durante questa fase l'individuo viene valutato sulla somiglianza rispetto al modello e sulla sua capacità di risolvere il problema. Queste due fitness vengono sommate e pesate nel seguente modo:

$$fit_{tot} = fit_{imit} \cdot (1 - \alpha) + fit_{perf} \cdot \alpha \tag{4.1}$$

$$\alpha = \frac{c}{N} \tag{4.2}$$

dove fit_{imit} e fit_{perf} sono rispettivamente le fitness sul problema di imitazione e categorizzazione, α il peso, $0 < c \leq N$ il numero del ciclo e $N = 350$ la durata della fase. Attraverso questo andamento di α l'agente viene spinto a migliorare la sua strategia da un punto di vista del problema in esame facendo in modo che essa sia un affinamento della strategia appresa piuttosto che una nuova strategia.

Facendo seguire una fase di apprendimento individuale alla fase di apprendimento sociale senza l'ausilio di α abbiamo riscontrato una tendenza degli agenti ad apprendere una nuova strategia che non prendesse in considerazione ciò che era stato appreso durante la fase sociale. In questo modo il processo non tiene in considerazione di quanto svolto dall'apprendimento sociale vanificandone la funzione ed il costo, degenerando in una fase di apprendimento individuale di una durata minore rispetto alla durata dell'apprendimento individuale effettuata dal modello (portando quindi ad una fitness finale più bassa). I protocomportamenti generati dalla fase di apprendimento verrebbero in qualche modo "sovrascritti" dalla nuova strategia piuttosto che sviluppati.

Abbiamo eseguito un secondo esperimento, basato sul setup sperimentale variato esposto precedentemente, in cui una popolazione di 50 individui viene fatta apprendere una strategia adattiva, ottenuta con 500 cicli di apprendimento individuale esposto nel capitolo precedente, attraverso l'algoritmo di apprendimento sociale appena descritto con l'aggiunta della seconda fase di apprendimento misto pesato.

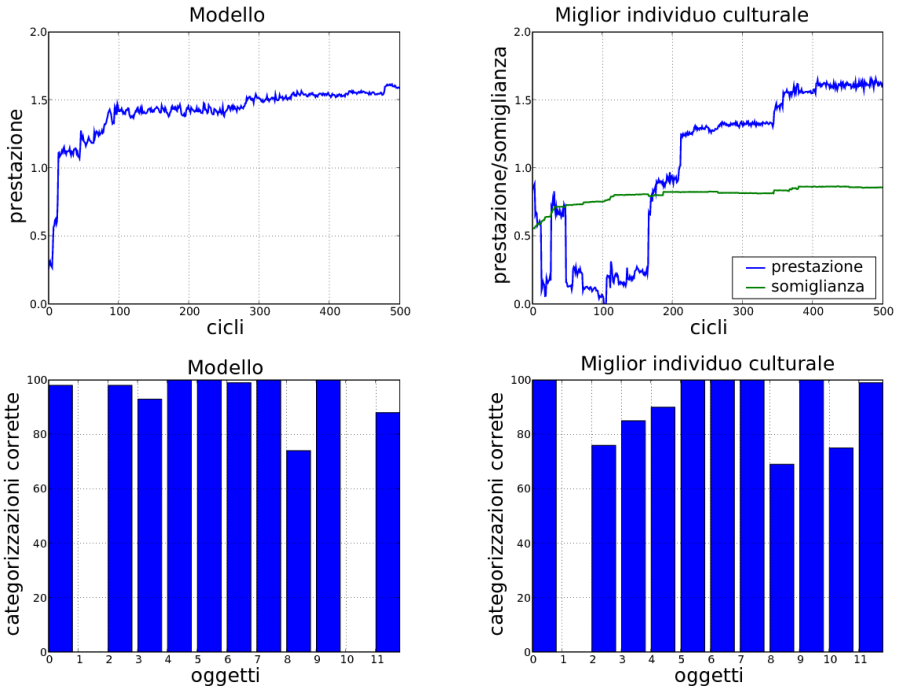


Figura 4.5: Andamento dell'apprendimento e categorizzazione del modello e del miglior individuo che apprende attraverso l'algoritmo di apprendimento sociale.

I parametri utilizzati sono gli stessi descritti per il setup sperimentale del capitolo precedente e cioè: 500 cicli di iterazione, 50 individui $e_{min} = 20, e_{max} = 60$, tasso di mutazione del 2%.

	Prestazione Modello	Prestazione Studente
Migliore	1.5883	1.6550
Media	1.2437	1.3151

Tabella 4.2: Prestazioni della popolazione da cui è estratto il modello e della popolazione che apprende attraverso l'algoritmo di apprendimento sociale. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 100 epoche.

Come si può notare nella tabella 4.2, la popolazione di individui che apprendono socialmente ottiene una fitness media più elevata rispetto alla media della popolazione di individui che apprendono individualmente da cui il modello è estratto. Ciò significa che esiste comunicazione nella fase di apprendimento sociale, che questa viene sfruttata dalla susseguente fase di apprendimento individuale e che questa comporta un innalzamento della fitness media. Si noti inoltre che esiste almeno un individuo della popolazione di individui che apprendono socialmente la cui fitness supera quella del modello. Questo significa che l'agente è riuscito a sfruttare le informazioni apprese durante la fase di apprendimento sociale per migliorare la strategia del modello. Analizzando la

categorizzazione (in figura 4.5) e i comportamenti del modello con quelli di questi agenti e possibile notare che il miglioramento avviene a più livelli:

- a livello della categorizzazione: un oggetto categorizzato correttamente in più (il modello categorizzava correttamente 10 oggetti mentre il miglior studente ne categorizza 11).
- a livello di comportamento motorio: l'agente raggiunge o si allontana dall'oggetto più velocemente attraverso traiettorie generalmente più "smooth".
- a livello globale: un oggetto categorizzato correttamente in più e migliore categorizzazione degli oggetti già categorizzati correttamente.

La capacità di apprendere un comportamento da un altro agente apre una serie di considerazioni sulle caratteristiche "divide-and-conquer" di questi algoritmi affrontate nella sezione 2.2.

Una prima considerazione riguarda la relazione tra imitazione e implementazione di una soluzione. Una volta appurato che un comportamento possa essere appreso da un altro agente, si può considerare la scelta di un modello che possiede la skill richiesta e la sua imitazione come l'implementazione di tale sottoproblema. Questo genere di trasferimento di informazione, a differenza del cross-over che richiede che nel genotipo le strategie siano implementate in gruppi di geni adiacenti e funzionalmente chiusi, non è legato alla rappresentazione genotipica o architetturale del sistema di controllo.

Una seconda considerazione riguarda il numero di individui da cui si può apprendere. Con la riproduzione sessuata, dove il genotipo del figlio è una mistura dei genotipi dei due genitori, il numero di agenti da cui si possono ottenere informazioni è limitato a due (i genitori) mentre nel caso dell'apprendimento sociale il numero è illimitato. Questo significa che potenzialmente un agente può apprendere e ricomporre molteplici sottosoluzioni (ognuna appresa da un modello diverso) per generare comportamenti globali più complessi.

I vantaggi adattivi ottenuti dalla combinazione di adattamento sociale ed individuale possono essere spiegati nei termini dell'effetto che hanno sull'esplorazione dello spazio dei parametri. Volendo effettuare un parallelo con l'adattamento evolutivo, si può affermare che entrambe le strategie si basano su una combinazione di due tipi di esplorazione: una che avviene ad alto livello che seleziona la zona da esplorare ed una a più basso livello che affina la ricerca nell'area selezionata precedentemente. Nell'adattamento evolutivo l'esplorazione ad alto livello viene effettuata dalla selezione dei padri da cui vengono generati gli N figli, attraverso la mutazione. Tutti i figli condividono buona parte del codice genetico tranne la parte mutata, che mediamente si attesta attorno al 5%, e quindi esplorano la stessa zona dello spazio dei parametri anche se in punti diversi. Le due fasi sono quindi legate dalla stessa rappresentazione e sono due operatori che agiscono direttamente sui parametri.

Anche nell'adattamento sociale l'esplorazione avviene a due livelli ma il primo, quello che seleziona la zona da esplorare più finemente, non è legato direttamente ai parametri di un individuo padre (da cui li eredita tramite il processo riproduttivo). Il modello, come il genitore, definisce un prototipo di comportamento da affinare da parte degli studenti e rappresenta un punto nello spazio dei comportamenti. Il fatto che la somiglianza tra il modello e lo studente sia

nello spazio dei comportamenti (e non nello spazio dei parametri) può in linea di principio consentire un'esplorazione dello spazio dei parametri che sia meno soggetto ai minimi locali di cui esso può essere caratterizzato. La prima selezione non avviene a livello dei parametri e della fitness sul problema e non è quindi influenzata da eventuali minimi locali (nello spazio dei parametri) in cui il modello potrebbe trovarsi. La fase finale di adattamento individuale tuttavia affina, a livello di spazio dei parametri, la soluzione esattamente come avviene attraverso la mutazione nell'adattamento evolutivo.

Questo tipo di considerazioni nascono dalla relazione che intercorre tra lo spazio dei parametri, cioè lo spazio in cui l'algoritmo ricerca le soluzioni, e lo spazio dei comportamenti, cioè lo spazio delle possibili strategie che possono emerge dall'embodiement dei parametri del sistema di controllo nel robot all'interno dell'ambiente. Esistono più punti all'interno dello spazio dei parametri che corrispondono ad un medesimo punto (ed il suo intorno) nello spazio dei comportamenti. La funzione della fase di apprendimento sociale è quindi quella di usufruire di questa relazione esplorando lo spazio dei parametri in aree diverse, che corrispondano però allo stesso punto nello spazio dei comportamenti, in cerca di morfologie prive di minimi locali.

Le considerazioni che vedono le caratteristiche complementari dell'apprendimento sociale ed individuale di apprendere, rispettivamente, un macrocomportamento attraverso una divisione in sottoproblemi e la ricerca ed implementazione delle rispettive sottosoluzioni trovano riscontro nelle osservazioni in animali reali [18, 17], dove l'apprendimento sociale ed individuale sono sempre accoppiati e interconnessi.

Esperimento 3: Apprendimento sociale e individuale esteso

Abbiamo quindi eseguito un terzo esperimento in cui abbiamo fatto apprendere individualmente una popolazione per 1000 cicli per capire se vi sia un vantaggio computazionale nell'utilizzare l'apprendimento sociale per 500 cicli a seguito della creazione di un modello attraverso l'algoritmo di apprendimento individuale (anch'esso per 500 cicli, 1000 quindi in tutto) rispetto al solo apprendimento individuale per 1000 cicli.

	Performance Modello c=1000	Performance Studente c=500+500
Migliore	1.6883	1.6550
Media	1.3788	1.3151

Tabella 4.3: Prestazioni della popolazione che apprende individualmente per 1000 cicli e della popolazione che apprende attraverso l'algoritmo di apprendimento sociale. I valori riportati si riferiscono alle prestazioni delle soluzioni ottenute alla fine del processo di adattamento stimate su 100 epoche.

I risultati in tabella 4.3 mostrano come l'apprendimento individuale per 1000 cicli porti ad una fitness media e massima maggiore, in entrambi i casi, rispetto all'utilizzo dell'apprendimento sociale.

Questi risultati mostrano come il modello di imitazione utilizzato non sia abbastanza potente e ricco di informazione da portare ad un vantaggio in termini di rapporto *costo computazionale/fitness*. Tale risultato sembra dunque suggerire che la combinazione di apprendimento sociale e individuale posso portare

ad un miglioramento delle prestazioni solo in quei casi in cui il tempo di apprendimento sia limitato. È tuttavia anche possibile ipotizzare che il modello di apprendimento misto proposto presenti delle caratteristiche errate che possano essere corrette che ne riducono le potenzialità dal punto di vista algoritmico.

4.4 Analisi dei risultati

Il vantaggio della funzione di somiglianza descritta nella sezione 4.2 è quello di permettere di descrivere la somiglianza tra due agenti nei termini più generali possibili; infatti non vi è nessun legame tra la stessa, la funzione di fitness, il problema e la strategia adattiva da imitare. A differenza della funzione di fitness infatti, la funzione di somiglianza precedentemente descritta è generale al sistema embodied e situated inquanto indipendente dall'architettura del sistema di controllo, dal numero e tipo di sensori ed attuatori, dal numero di agenti presenti nell'ambiente e dalle caratteristiche dell'ambiente stesso.

Analizzando l'andamento delle curve di somiglianza durante la fase mista di apprendimento sociale ed individuale si può notare che esse non presentano un drastico calo che intuitivamente ci potremmo aspettare considerando il fatto che in tale fase l'adattamento viene guidato dalla funzione di fitness e che quindi la strategia adottata possa essere variata sensibilmente. Un altro indicatore dell'inefficienza della funzione di somiglianza è la relazione tra le categorizzazioni ed il fattore di somiglianza. Se è vero che categorizzazioni simili non necessitano necessariamente di strategie simili è altresì vero che un fattore di somiglianza alto preveda una categorizzazione simile. Analizzando tuttavia le categorizzazioni in figura 4.6 dell'agente con massimo fattore di somiglianza rispetto al modello non abbiamo riscontrato categorizzazioni così simili rispetto allo stesso (come avviene ad esempio con l'individuo con massima performance in figura 4.5.

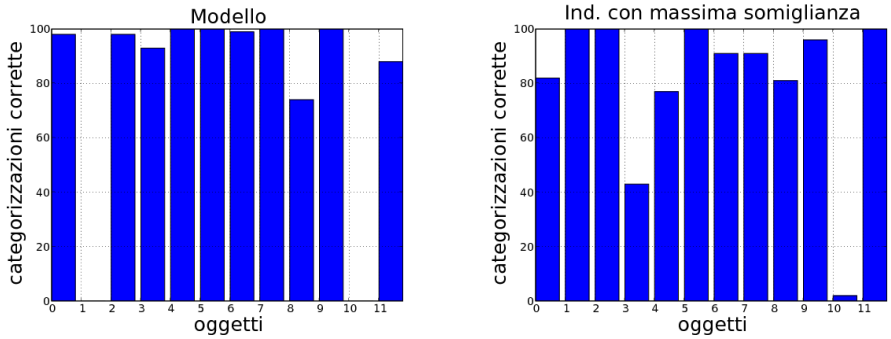


Figura 4.6: Categorizzazioni del modello e dell'individuo estratto dalla popolazione di individui che apprendono attraverso l'algoritmo di apprendimento sociale. Si noti che, nonostante l'individuo sulla destra sia l'individuo con il massimo fattore di somiglianza alla fine dell'apprendimento, la sua categorizzazione è più differente da quella del modello rispetto alla categorizzazione dell'individuo con massima performance esposto in figura 4.5

Sarebbe proprio la generalità della funzione di somiglianza la limitazione all'approccio descritto all'apprendimento sociale. Si consideri ad esempio un agente studente che sviluppa un comportamento uguale ma speculare a quello

del modello. Lo studente avrebbe uguale capacità di categorizzazione degli oggetti ma avrebbe un comportamento motorio speculare (approccerebbe l'oggetto con la stessa spirale del modello ma girando verso destra rispetto ad un modello che effettua la stessa spirale ma in senso contrario). Per la funzione di somiglianza tale comportamento riceverebbe un bassissimo fattore di somiglianza nonostante il comportamento sarebbe fortemente simile nei termini della strategia adattiva al problema. Allo stesso modo un approccio all'oggetto con una spirale più stretta rispetto ad una spirale larga effettuata dal modello risulterebbe ottenere un fattore di somiglianza basso inquanto l'output degli attuatori sarebbero proporzionalmente diversi.

Una delle strategie adattive sviluppate con il setup sperimentale proposto prevedeva che il robot eseguisse una spirale avvicinando l'oggetto e curvasse per allotanarlo nel caso esso fosse categorizzato come cattivo o vi si fermasse vicino nel caso fosse categorizzato come buono. Analizzando gli output degli attuatori si è potuto notare che il robot, seppur eseguisse una spirale dotata di una buona "smoothness", presentava un forte andamento oscillatorio irregolare sugli attuatori (come si può notare in figura 2.4). Inoltre, nel caso degli oggetti categorizzati come buoni, la strategia adottata di stare fermo sul posto era anch'essa caratterizzata da un continuo andamento oscillatorio irregolare. Questo tipo di strategia sarebbe da imputare alla natura embodied e situated del sistema ed alle caratteristiche del sistema visivo implementato nell'agente. Il fatto che l'agente vedesse alternativamente da uno dei 2 neuroni (essi venivano attivati entrambi solo quando l'oggetto era esattamente di fronte all'agente) a seconda che l'oggetto fosse nel campo visivo destro o sinistro dell'agente, ha portato l'algoritmo di apprendimento individuale a sviluppare una strategia oscillatoria come quella appena descritta. Un robot che si comportasse esattamente come il robot modello, da un punto di vista della traiettoria adottata, ma che non presentasse un andamento oscillatorio negli attuatori risulterebbe non simile a tale agente.

Un altro possibile esempio è quello di un robot che deve raggiungere un punto B partendo da un punto A. Si supponga che due strategie vengano sviluppate dal sistema e che entrambe prevedano di raggiungere il punto B attraverso la linea retta che congiunge i due punti. Tuttavia la prima strategia prevede di raggiungere la destinazione procedendo in "retromarcia" mentre la seconda strategia prevede di raggiungerla procedendo in modo frontale. Nuovamente la funzione di somiglianza considererebbe i due comportamenti completamente differenti nonostante dal punto di vista dell'osservatore esterno così come dal punto di vista del valore adattivo delle due strategie esse siano molto simili.

Sembrerebbe quindi che il problema imitativo sia troppo restrittivo e complesso per apportare un vantaggio rispetto al trial & error che riceve invece un feedback diretto sul problema da risolvere.

Queste considerazioni suggeriscono che l'aspetto di debolezza dell'algoritmo proposto consiste nella misura usata per implementare l'apprendimento sociale. Delle misure più indirette, rispetto all'imitazione delle singole azioni, basate per es. sulla somiglianza tra gli effetti dei comportamenti esibiti, potrebbero dunque rivelarsi più efficaci.

4.5 Sommario

In questo capitolo abbiamo mostrato come l'utilizzo di una funzione obiettivo, detta funzione di somiglianza, applicata all'algoritmo esposto nel capitolo precedente permetta ad un agente di apprendere un comportamento esibito da un altro agente esperto. Inoltre abbiamo mostrato come la sola minimizzazione di tale funzione non sia sufficiente all'acquisizione di un comportamento adattivo ma che essa debba essere affiancata da una massimizzazione della funzione di fitness. Abbiamo inoltre mostrato come un algoritmo misto basato su apprendimenti individuale e sociale sia vantaggioso solo quando il tempo di apprendimento sia limitato. Infine sulla base dei risultati ottenuti abbiamo discusso quali possibili variazioni dell'algoritmo proposto potrebbero portare a risultati più interessanti dal punto di vista algoritmico.

Capitolo 5

Conclusioni

In questa tesi abbiamo mostrato alcuni approcci all'adattamento emergente ed auto-organizzante in sistemi embodied e situated.

Nel capitolo 3 abbiamo mostrato come il Simulated Annealing e gli algoritmi genetici possano essere utilizzati per far emergere un comportamento adattivo al problema in esame. Abbiamo inoltre mostrato come una variazione dell'algoritmo standard che prevede una ri-valutazione delle prestazioni della soluzione corrente, un aumento progressivo del tempo in cui viene stimata la fitness della soluzione corrente, e una temperatura nulla produce risultati migliori sia dal punto di vista delle prestazioni raggiunge che dal punto di vista del costo computazionale.

Nel capitolo 4 abbiamo mostrato come l'efficacia di un apprendimento individuale possa essere aumentata permettendo agli individui di apprendere in interazione con altri individui già sottoposti ad un processo di apprendimento. Questo risultato è stato ottenuto introducendo una funzione che descrive le differenze a livello senso-motorio e minimizzando tale funzione attraverso l'algoritmo descritto nel capitolo 3. Abbiamo inoltre mostrato che la minimizzazione di tale funzione non è sufficiente per acquisire il comportamento esibito dal modello, ma che essa debba essere affiancata dalla massimizzazione della funzione di fitness. I risultati ottenuti hanno mostrato come tale forma di apprendimento sociale possa consentire una velocizzazione del processo di apprendimento. Infine sulla base dei risultati ottenuti abbiamo discusso quali variazioni dell'algoritmo proposto possano portare a vantaggi anche rispetto alle prestazioni raggiunte alleviando il problema della convergenza su soluzioni che rappresentano dei minimi locali nello spazio di ricerca.

Appendice A

Implementazione

Di seguito viene riportato il codice riguardante l'implementazione dei due algoritmi proposti, utilizzato nei capitoli precedenti per la risoluzione del problema adattivo scelto.

Per l'implementazione dell'algoritmo di apprendimento individuale proposto:

```
void
individual_learning()
{
    int i,c,n;
    int *cgeno;
    int bkg[ind->numchars];
    int evaluatedn[nindividuals], nrejected[nindividuals];
    double newperformance, bestperf[nindividuals], totperformance[nindividuals];
    int bestind = 0;
    double bestfit = 0;

    // init arrays:
    memset(evaluatedn, 0, nindividuals*sizeof(int));
    memset(nrejected, 0, nindividuals*sizeof(int));
    for(n=0; n < nindividuals; n++)
        bestperf[n] = -1;

    for(c=0; c < learningc; c++){
        amoebaepochs = (int) amoebae_min + (c*(amoebae_max - amoebae_min)/learningc);
        for(n=0; n < nindividuals; n++) {
            // backup genotype
            for(i=0, cgeno=*(genome+n); i < ind->numchars; i++, cgeno++)
                bkg[i] = *cgeno;

            // perturbe parameters
            for (i=0, cgeno=*(genome+n); i < ind->numchars; i++, cgeno++)
                *cgeno = mutate(*cgeno);

            // evaluate the individual
            newperformance = eval_perf(c, n);

            // if performance increased
            if(newperformance >= bestperf[n]){
                bestperf[n] = newperformance;
                evaluatedn[n] = 1;
                nrejected[n] = 0;
            } else {
                for(i=0, cgeno=*(genome+n); i < ind->numchars; i++, cgeno++)
                    *cgeno = bkg[i];
                nrejected[n]++;
                bestperf[n] = (bestperf[n]+eval_perf(c, n))/2;
            }
        }// end of learningcycle for all the individuals
    }
}
```

Per l'algoritmo di apprendimento sociale proposto:

```
void
cultural_learning()
{
    int i, g, n, k, nrejected = 0;
    double err, lasterr, last_im_perf;
    float w1, w2, weight_d; // mixed phase's weights
    double newperf, lastperf, new_m_perf, last_m_perf, last_fi_perf;
    int oldgen[ind->numchars];
    char sbuffer[256];
    int *cgeno;

    for(g=1; g < generations; g++) {
        randomize_pop(); // starting from initial random solutions

        for(n=0; n < nindividuals; n++) {
            // select individual to imitate
            int m = select_model(0);

            lasterr = newperf = lastperf = last_m_perf = new_m_perf = 0;
            last_im_perf = last_fi_perf = 0;
            int type = 1;
            w1 = 1;
            w2 = 0;

            for(i = 0; i < imitatingc+learningc; i++) {
                // linear progression of epochs number
                amoebaepochs = (int) amoebae_min + (i*(amoebae_max - amoebae_min)/(imitatingc+learningc));
                // backup genotype
                for(k=0, cgeno=*(genome+n); k < ind->numchars; k++, cgeno++)
                    oldgen[k] = *cgeno;

                // perturbe parameters
                for (k=0, cgeno=*(genome+n); k < ind->numchars; k++, cgeno++)
                    *cgeno = mutate(*cgeno);

                // evaluate social performances
                err = imitate(i, n, type);

                // if in second phase, test problem performance too
                if(i > imitatingc){
                    weight_d = ((float)(i-imitatingc)/learningc);
                    w1 = (1 - weight_d);
                    w2 = weight_d;
                    newperf = eval_perf(i, n);
                }

                new_m_perf  = err*w1 + newperf*w2;
                last_m_perf = last_im_perf*w1 + last_fi_perf*w2;

                if(new_m_perf < last_m_perf){// refused
                    for(k=0, cgeno=*(genome+n); k < ind->numchars; k++, cgeno++)
                        *cgeno = oldgen[k];
                    nrejected++;

                    // retest
                    last_im_perf = (last_im_perf + imitate(g, n, type))/2;
                    if(i > imitatingc)
                        last_fi_perf = (last_fi_perf + eval_perf(g, n))/2;
                    last_m_perf  = last_im_perf*w1 + last_fi_perf*w2;
                } else {
                    nrejected = 0;
                    last_m_perf  = new_m_perf;
                    last_im_perf = err;
                    last_fi_perf = newperf;
                }
            }
        }
        select_models(g); // select next generation's models
    }
}
```

Bibliografia

[1] S. Nolfi A. Acerbi, D. Marocco. Social facilitation on the development of foraging behaviors in a population of autonomous robots. *Almeida e Costa, F. et al. (Eds.), Advances in Artificial Life. Proceedings of ECAL 2007*, pages 625–634, 2007.

[2] S Acerbi, A.; Nolfi. Social learning and cultural evolution in embodied and situated agents. *Artificial Life IEEE Symposium*, 1-5:333 – 340, 2007.

[3] J.T. Alander. Indexed bibliography of genetic algorithms in operations research. *Report 94-1-OR, Department of Information Technology and Production Economics, University of Vassa, Finland*, 1995.

[4] R.C. Arkin. *Behavior-based Robotics*. MIT Press, 1998.

[5] Rogers A.S. Does biology constrain culture? *American Anthropologist*, 90:819–831, 1989.

[6] R. D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1-2):173–216, January 1995.

[7] R.D. Beer. The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11(4):209–243, 2003.

[8] R. Boyd and Richerson P.J. *Culture and the evolutionary process*. Chicago: The University of Chicago Press, 1985.

[9] R. Boyd and P. J. Richerson. Why does culture increase human adaptability? *Ethology and Sociobiology*, 16:125–143, 1995.

[10] Richerson P. Boyd R. Why culture is common, but cultural evolution is rare. 1996.

[11] R. A. Brooks. Intelligence without reason. 1991.

[12] R.A. Brooks. *Cambrian Intelligence*. MIT Press, 1999.

[13] P. Cerveri, A. Pedotti, and N.A. Borghese. Combined evolution strategies for dynamic calibration of video-based measurement systems. *Evolutionary Computation, IEEE Transactions on*, 5:3:271–282, 2001.

[14] J. Demiris and G. Hayes. Active and passive routes to imitation. In *Proceedings of the AISB'99 Symposium of Imitation in Animals and Artifacts*, 1999.

[15] G. di Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Understanding motor events: a neurophysiological study. *Exp Brain Res*, 91(1):176–180, 1992.

[16] Baldassarre G., Parisi D., and Nolfi S. Distributed coordination of simulated robots based on self-organisation. *Artificial Life*, (12) 3:289–311, 2006.

[17] B. G. Galef. *Introduction: social learning and imitation in Social learning and imitation: The roots of culture.* C. M Heyes and B. G. Galef, Eds., New York: Academic Press, 1996.

[18] B. G. Galef and K. N. Laland. Social learning in animals: empirical studies and theoretical models. *BioScience*, 55:6:489–500, 2005.

[19] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Professional, 1989.

[20] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[21] T. Kameda and D. Nakanishi. Cost-benefit analysis of social/cultural learning in a non-stationary uncertain environment: An evolutionary simulation and an experiment with human subjects. *Evolution and Human Behavior*, 23:373–393, 2002.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[23] C. Koulamas, S. R. Antony, and R. Jaen. A survey of simulated annealing applications to operations research problems. *Omega International Journal of Management Science, Omega*, Vol. 22 No.1:41–56, 1994.

[24] Oliver J.H. Tu W. Malhotra, A. Synthesis of spatially and intrinsically constrained curves using simulated annealing. *ASME Journal of Mechanical Design*, 118:53–61, 1996.

[25] M. Mitchell. *An Introduction to genetic algorithms.* MIT Press, 1996.

[26] R. Murray-Smith and T. A. Johanson. *Multiple model approaches to modelling and control.* Taylor and Francis, 1998.

[27] Jason Noble and Peter M. Todd. Imitation or something simpler? modeling simple mechanisms for social information processing. pages 423–439, 2002.

[28] S. Nolfi. Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour. *ComPlexUs*, (2):195–205, 2004/2005.

[29] S. Nolfi and D. Floreano. *Evolutionary robotics.* MIT Press, 2000.

[30] S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, pages 5:99–105, 1997.

[31] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, 1998.

[32] Dawkins R. The selfish gene. 1976.

[33] G.R. Reynolds. An introduction to cultural algorithms. *In Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 131–139, 1994.

[34] R.A. Rutenbar. Simulated annealing algorithms: an overview. *Circuits and Devices Magazine, IEEE Volume 5, Issue 1, Page(s):19 - 26*, 1989.

[35] Schaal. Is imitation learning the route to humanoid robots? *Trends Cogn Sci*, 3(6):233–242, Jun 1999.

[36] S. Schaal. Learning from demonstration. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1040. The MIT Press, 1997.

[37] Lemahieu I. Sundermann, E. Pet image reconstruction using simulated annealing. *Medical Imaging 1995: Image Processing*, pages 378–386, 1995.

[38] J. Terkel. Cultural transmission of feeding behaviour in the black rat (rattus rattus). *C. M. Heyes and B. G. Galef, Jr., editors. Social Learning in Animals: The Roots of Culture. Academic*, pages 17–47, 1996.

[39] P.J. van Laarhoven and E.H. Aarts. *Simulated Annealing: Theory and Applications (Mathematics and Its Applications)*. Springer, 1987.

[40] F.J. Varela, E. Thompson, and E. Rosch. *The Embodied Mind: Cognitive Science and Human Experience*. Cambridge, MA: MIT Press, 1991.

[41] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11:1317–1329, 1998.