Alma Mater Studiorum Università di Bologna
Campus di Cesena

SCUOLA DI INGEGNERIA
Corso di Laurea in Ingegneria e Scienze informatiche

Adattamento online di robot controllati da reti booleane

Tesi in
Sistemi Intelligenti Robotici

Relatore:
Prof. Andrea Roli

Presentata da:
Edoardo Barbieri

Co-relatore:
Dott. Michele Braccini

IV Sessione di laurea
Anno Accademico 2019/2020

Indice

Introduzione

In questa tesi si esplora la possibilità di usare reti booleane come software di controllo per sistemi robotici. Ogni robot viene controllato da una rete booleana generata casualmente. Mappando alcuni nodi della rete ai sensori e agli attuatori presenti, essa diventa il nucleo computazionale del robot. Il comportamento dell'agente è quindi determinato dalle dinamiche prodotte dalla rete booleana. In questa tesi si vuole determinare se le reti booleane casuali possono essere usate con successo come software di controllo, senza doverne modificare la struttura attraverso un procedimento evolutivo. Le tecniche di adattamento esposte in seguito sono state definite per esplorare questa ipotesi. Una tecnica prevede soltanto di alterare la mappatura tra i nodi della rete e i sensori e gli attuatori presenti sul robot, cercando di sfruttare al meglio le capacità computazionali offerte dalla rete booleana, senza doverne modificare le caratteristiche. Un'altra, essendo simile ad un processo evolutivo, prevede di alterare la struttura interna della rete. Queste due tecniche poi verranno confrontate in termini di prestazioni. Il processo di adattamento è svolto "online", infatti, le tecniche vengono applicate mentre i robot sono in azione nell'ambiente. Inoltre, ci si concentra fortemente sui parametri di generazione delle reti booleane, in quanto, in base a questi è possibile ottenere tre regimi di funzionamento distinti: ordinato, critico, caotico. Queste tre tipologie di reti booleane vengono testate per ogni tecnica di adattamento, e vengono messe a confronto sulla base dei risultati ottenuti dai robot. Quest'ultimi sono immersi in ambienti virtuali in cui devono cercare di massimizzare una determinata funzione obiettivo. La bontà del software di controllo viene misurata sulla base del punteggio che il robot produce. L'ipotesi è che i robot dotati di una rete booleana con un regime di funzionamento critico non necessitano di modificare la struttura della rete per massimizzare il punteggio. Infatti, questi devono soltanto trovare un interfacciamento ottimale con essa. Invece, le reti ordinate e caotiche potrebbero beneficiare di un'evoluzione attraverso una riconfigurazione della struttura interna.

Successivamente, viene studiata la robustezza delle reti booleane rispetto a perturbazioni e malfunzionamenti introdotti artificialmente. Infine, viene effettuata una breve analisi sulla relazione tra il punteggio ricavato dalla funzione obiettivo e l'entropia prodotta dai sensori del robot. Lo scopo di quest'ultima analisi è di provare a guidare l'adattamento attraverso delle misure di entropia che sono indipendenti dalla funzione obbiettivo.

I capitoli sono suddivisi in questo modo:

- Capitolo 1: si introduce il modello delle reti booleane. In particolare, si tratta delle reti booleane casuali, e di come queste possano essere generate per ottenere tre distinti regimi di funzionamento: ordinato, critico, caotico.

- Capitolo 2: si descrive l'apparato tecnico degli esperimenti compiuti, ovvero, il simulatore utilizzato, gli ambienti virtuali definiti, i robot schierati e le funzioni obiettivo. Inoltre, si elencano le tecniche di adattamento utilizzate ed il loro impiego nelle simulazioni. Infine, si descrive l'obiettivo di ricerca dei vari esperimenti, indicando quali informazioni si vogliono ottenere.

- Capitolo 3: si mostrano i risultati raccolti dagli esperimenti, discutendo degli effetti delle tecniche di adattamento e dei regimi di funzionamento delle reti booleane. Successivamente, si esamina l'evoluzione delle reti booleane casuali sottoposte a mutazione.

- Capitolo 4: si tratta della robustezza alle perturbazioni delle reti booleane. In particolare, si testano due tipologie di perturbazioni: una continua durante la fase di adattamento, e una istantanea sulle reti booleane già evolute. Infine, si studia il comportamento delle reti booleane sottoposte a malfunzionamenti strutturali.

- Capitolo 5: si studia il comportamento dei robot usando una misura tipica della teoria dell'informazione. Si mostra la relazione tra *fitness* ed entropia. Infine, si propone una funzione obiettivo che sfrutti questa relazione, con l'intento di conferire ai robot istinti utili allo sviluppo dei comportamenti desiderati.

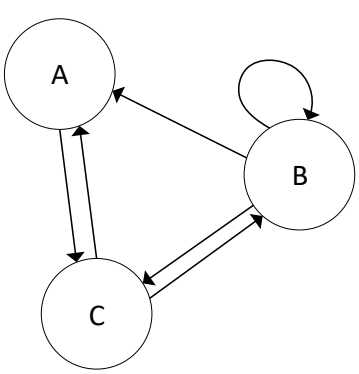
Capitolo 1

Reti booleane

Le reti booleane o *Boolean Networks* (BN) sono state introdotte da Kauffman [7] alla fine degli anni Sessanta. Sin dalla loro definizione si sono dimostrate utili in contesti biologici, in particolare, sono state impiegate per la modellazione di reti di regolazione genetica. Le reti booleane sono un esempio di sistemi dinamici complessi e possono essere impiegate in ambiti che si discostano dal dominio biologico, infatti, vi è un particolare interesse anche in contesti computazionali. Esse offrono un'interessante prospettiva dal punto di vista ingegneristico siccome possono essere riprodotte dinamiche complesse pur avendo una definizione del modello molto compatta.

Formalmente una rete booleana è un sistema a stati discreti che avanza con una dinamica temporale discreta. La struttura di una rete booleana è rappresentabile da un grafo orientato con N nodi. Ogni nodo è costituito da una variabile booleana x_i e da una funzione booleana $f_i(x_{i_1}, ..., x_{i_{K_i}})$ dove x_{i_n} sono i nodi predecessori e K_i la cardinalità dei nodi entranti (grado entrante). Lo stato del sistema in un determinato istante $t, t \in \mathbb{N}$ è composto dall'insieme di variabili booleane $s(t) = \{x_i, ..., x_N\}$. Per determinare lo stato $s(t+1)$ è necessario conoscere lo stato predecessore $s(t)$: si calcola il nuovo stato di ogni variabile x_i usando la funzione f_i con i valori $\{x_{i_1}, ..., x_{i_{K_i}}\}$ dello stato $s(t)$. Questo particolare algoritmo di aggiornamento della rete booleana è sincrono e completamente deterministico. Una rete booleana composta da N nodi può avere un massimo di 2^N stati distinti. In figura 1.1 è mostrato un esempio di rete booleana con 3 nodi.

Il capitolo è basato sull'introduzione alle reti booleane nella tesi di dottorato di Michele Braccini [3].



B	C	A
0	0	1
0	1	1
1	0	0
1	1	1

B	C	B
0	0	0
0	1	0
1	0	1
1	1	1

A	B	C
0	0	0
0	1	0
1	0	1
1	1	1

Figura 1.1: Esempio di rete booleana composta da tre nodi con due archi entranti ciascuno. Le tavole di verità costituiscono le funzioni di aggiornamento dei rispettivi nodi.

Inizializzando la rete booleana con uno stato casuale, e applicando ripetutamente la funzione di aggiornamento dei nodi in modo sincrono, si ottiene una sequenza di stati della rete. La sequenza, essendo potenzialmente illimitata, ricorre in alcuni stati precedentemente visitati, creando un ciclo continuo di stati che si ripete illimitatamente in assenza di perturbazioni [1] esterne. Questo ciclo continuo di stati viene chiamato *attrattore ciclico* se è composto da due o più stati distinti, *punto fisso* se il ciclo è composto da un solo stato. In figura 1.2 è mostrato un semplice esempio di attrattori riferiti allo schema della rete booleana in figura 1.1.

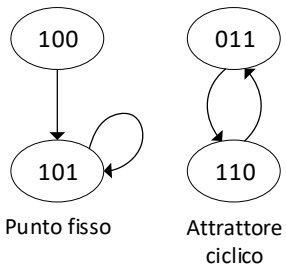


Figura 1.2: Esempio di sequenza di stati della rete booleana mostrata in figura 1.1. Partendo dallo stato $\{A = 1, B = 0, C = 0\}$ si incorre in un punto fisso. Partendo da $\{A = 0, B = 1, C = 1\}$ si incorre in un attrattore ciclico composto da una sequenza di due stati.

[1] Con perturbazione s'intende la negazione di una variabile arbitraria x_i tra uno stato della rete e il successivo.

1.1 Reti booleane casuali

Le reti booleane casuali o *Random Boolean Netowrks* (RBN) sono la prime ad essere state formulate. Esse vengono prodotte generando casualmente gli archi e le funzioni di aggiornamento dei nodi. Per ogni nodo vengono generate K connessioni entranti, dunque, i K vicini vengono scelti casualmente tra i restanti $N-1$ nodi. La funzione di aggiornamento di ogni nodo viene prodotta casualmente generando una tavola di verità con 2^K voci: viene definito un parametro p che determina la probabilità di assegnare il valore 1 a una voce della tavola di verità, viceversa, viene assegnato valore 0 con probabilità $1-p$.

Il comportamento di una RBN è fortemente influenzato dai parametri di generazione K e p. È possibile distinguere due regimi di funzionamento di una rete booleana: ordinato e caotico. Quest'ultimi vengono classificati in base alla propagazione delle perturbazioni, e dalla lunghezza e numero di attrattori della rete. Nelle reti ordinate una perturbazione in media viene propagata un numero di volte inferiore a uno. Questo significa che le perturbazioni svaniscono velocemente, facendo ricadere la rete booleana nell'attrattore precedente alla perturbazione. Infatti, le reti ordinate sono caratterizzate da un numero elevato di attrattori con una lunghezza molto corta. Nelle reti a regime caotico una perturbazione si propaga in media su più di un nodo, il cui effetto quindi non svanisce nel breve periodo. Gli attrattori di una rete caotica sono tipicamente molto lunghi ma di numero inferiore rispetto ad una rete ordinata.

È stata identificata un'equazione [1] che permette di definire il confine tra regimi ordinati e caotici sulla base dei parametri K e p. Le reti booleane casuali prodotte da una combinazione dei parametri che si trova sul confine delle due regioni sono dette critiche. Il confine è definito dall'equazione 1.1 e mostrato in figura 1.3.

$$K = \frac{1}{2p(1-p)} \tag{1.1}$$

Una perturbazione su una rete critica viene propagata in media ad un singolo nodo; l'effetto è che la rete riesce a memorizzare per un certo tempo l'informazione introdotta (perturbazione) senza lasciarla svanire.

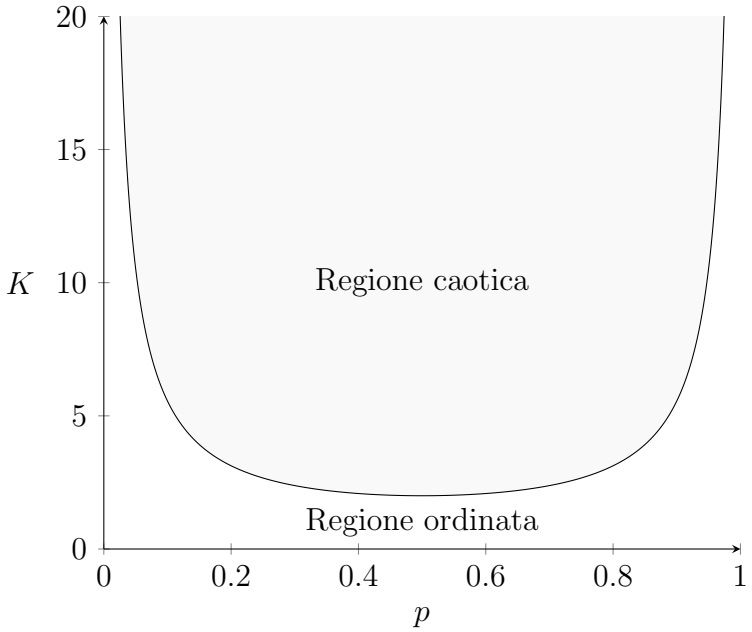


Figura 1.3: Rappresentazione del confine tra regione ordinata e caotica dei regimi di funzionamento delle RBN sulla base dei parametri di generazione K e p.

Le reti booleane in un regime critico mostrano i comportamenti più interessanti. Infatti, nella scienza dei sistemi complessi è presente da molto tempo una congettura (ipotesi di criticità / *criticality hypothesis*) in cui si afferma che i sistemi in un regime dinamico tra ordine e caos mostrano comportamenti caratterizzati da un equilibrio ottimale tra robustezza e adattabilità, con una capacità computazionale superiore [11]. Anche in questa tesi si vuole esplorare questa congettura, paragonando le differenti tipologie di reti booleane in termini di prestazioni ottenute dai robot. In questo caso gli agenti utilizzano le reti booleane come software di controllo. Si ipotizza che saranno proprio quei robot dotati di RBN generate con parametri sul confine tra ordine e caos a guadagnare vantaggio rispetto ai robot dotati di reti booleane ordinate o caotiche. Nel prossimo capitolo vengono descritte le tecniche adottate per testare e paragonare le diverse tipologie di reti booleane in un contesto robotico.

Capitolo 2

Descrizione tecnica

In questo capitolo sono esposti gli aspetti tecnici e tecnologici che hanno permesso di realizzare il progetto di tesi. S'introduce brevemente il simulatore adottato, per poi passare alla definizione delle nicchie ambientali in cui i robot vengono schierati. Si descrivono hardware e software dei robot, concentrandosi sull'utilizzo delle reti booleane e sulle tecniche di adattamento usate. Si espongono i vari task progettati e la codifica di essi sotto forma di funzioni obiettivo. Infine, sono elencati i parametri messi a confronto nelle varie simulazioni.

2.1 Simulatore: Argos3

Si è scelto di utilizzare il simulatore Argos3[1] in quanto si è dimostrato efficiente ed efficace in questo particolare caso di studio. Il simulatore permette di definire degli ambienti virtuali a propria discrezione attraverso un file di configurazione in cui sono specificate le varie caratteristiche: ostacoli, perimetro, terreno, fonti luminose e distribuzione dei robot. Nella prossima sezione vengono descritti ad alto livello i quattro ambienti definiti per questo progetto. Il simulatore permette di schierare in un ambiente uno o più robot andando a scegliere tra alcuni modelli fisici già prestabiliti, l'unica cosa da definire è il software di controllo che essi devono usare.

La scelta di utilizzare un simulatore piuttosto che portare avanti gli esperimenti in un ambiente reale è stata obbligata dall'elevato numero di test condotti. Questa scelta porta con sé alcuni svantaggi, come quello della discrepanza tra simulazione e realtà (*reality-gap*): è bene considerare che i risultati in un ambiente simulato possono differire da quelli che si sarebbero ottenuti utilizzando dei robot fisici. In questo progetto di tesi non vi è un particolare svantaggio nell'uso del simulatore, in quanto, l'obiettivo non è quello di addestrare dei robot per poi schierarli nel mondo reale, ma bensì, quello di confrontare alcune tecniche di adattamento, astraendo dai possibili difetti derivati dall'uso di un simulatore più o meno realistico. Un ringraziamento va agli sviluppatori di Argos3 per aver mantenuto il progetto *open-source*.

[1] Argos3: `https://www.argos-sim.info/`

2.2 Nicchie ambientali

Le simulazioni sono state effettuate in quattro ambienti distinti. Nella figura 2.1 sono mostrate le quattro possibili arene utilizzate. Tutti gli ostacoli sono statici e non possono essere spostati dai robot. Anche la sorgente luminosa della III e IV arena è fissata ad una posizione specifica. I robot, invece, vengono schierati in modo casuale, all'interno del perimento, sia per posizione che per rotazione iniziale.

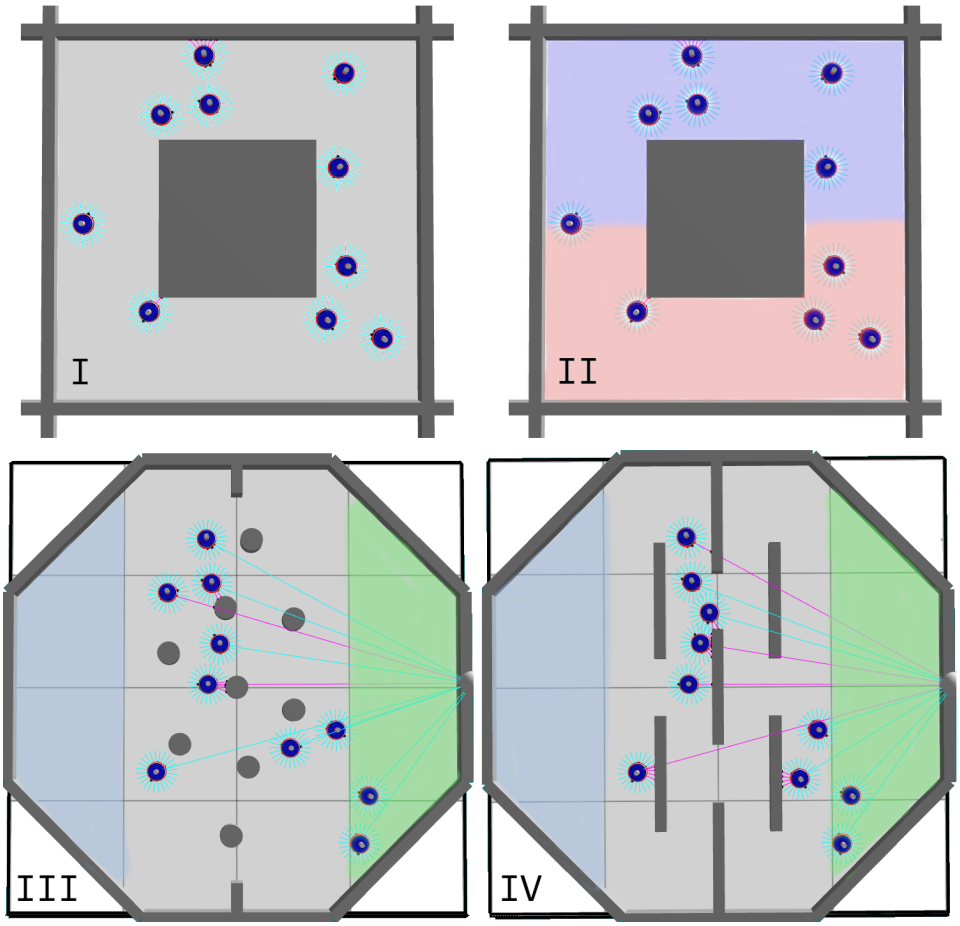


Figura 2.1: Visuale dall'alto delle quattro arene. Gli oggetti grigio scuro sono gli ostacoli, gli oggetti blu sono i robot. La prima e la seconda arena sono costituite da un perimetro rettangolare con un ostacolo quadrato al centro. La seconda arena si differenzia dalla prima per la definizione di due aree virtuali, rossa e blu, che serviranno per compiere un determinato task (descritto nella sezione 2.3.3). La terza e quarta arena sono caratterizzate da un perimetro ottagonale con svariati ostacoli al centro; le aree azzurre e verdi indicano due zone virtuali necessarie per la riuscita di un task di *foraging*. Le ultime due arene sono inoltre dotate di una sorgente luminosa posizionata sul confine esterno della zona verde.

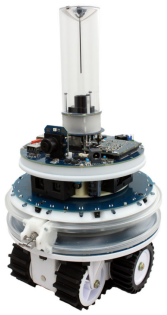
Per ogni arena è definito un task ben preciso, dunque, nei capitoli successivi si indica il numero dell'arena (I, II, III, IV) per discriminare sia l'ambiente che il task a cui ci si riferisce. Sono state sviluppate quattro arene distinte per avere una piattaforma di *benchmarking* più ampia, infatti, la difficoltà nell'eseguire il task è crescente dalla prima alla quarta arena. Questo permetterà di confrontare le tecniche di adattamento e la bontà del sistema di controllo in contesti differenti.

2.3 Robot

In questa sezione viene descritta la struttura hardware e software dei robot, i task che devono compiere e le tecniche di adattamento impiegate. In particolare, viene descritto in dettaglio l'uso delle reti booleane come software di controllo e le possibili forme di adattamento adottate.

2.3.1 Forma fisica

Si è scelto di usare il modello fisico del *foot-bot* [2] in quanto rappresenta un robot di piccole dimensioni, dal costo contenuto, ed equipaggiato di tutti i componenti (sensori / attuatori) necessari. Ogni robot utilizza i seguenti componenti:



- *differential steering*: attuatore che permette di controllare i motori delle due ruote motrici.
- *proximity sensors*: 24 sensori di prossimità intorno al robot che permettono di individuare oggetti vicini.
- *light sensors*: 24 sensori di luminosità intorno al robot che permettono di captare l'intensità di segnali luminosi.

2.3.2 Forma di controllo

Ogni robot utilizza una rete booleana come nucleo computazionale per il proprio sistema di controllo. Il comportamento manifestato dal robot dipende dalla dinamica interna della rete e da come essa è interconnessa ai sensori e agli attuatori. In figura 2.2 è esposto lo schema di come le reti booleane vengono usate dagli agenti. È bene fare una distinzione sui termini usati per riferirsi ad alcuni nodi della rete booleana:

Nodi di input : con questo termine si fa riferimento ad un particolare sottoinsieme di nodi di una rete booleana. In figura 2.2 sono rappresentati come nodi verde chiaro. Tali nodi vengono usati per perturbare lo stato

della rete con i segnali provenienti dai sensori del robot: dopo la fase di codifica, si sovrascrive lo stato dei nodi di input con i valori ricavati dai sensori.

Nodi di output : anche in questo caso ci si riferisce ad un sotto-insieme di nodi della rete booleana. In figura 2.2 sono rappresentati come nodi azzurro chiaro. Lo stato di questi nodi viene utilizzato per controllare gli attuatori del robot. Ogni nodo di output avrà un particolare effetto su un determinato attuatore.

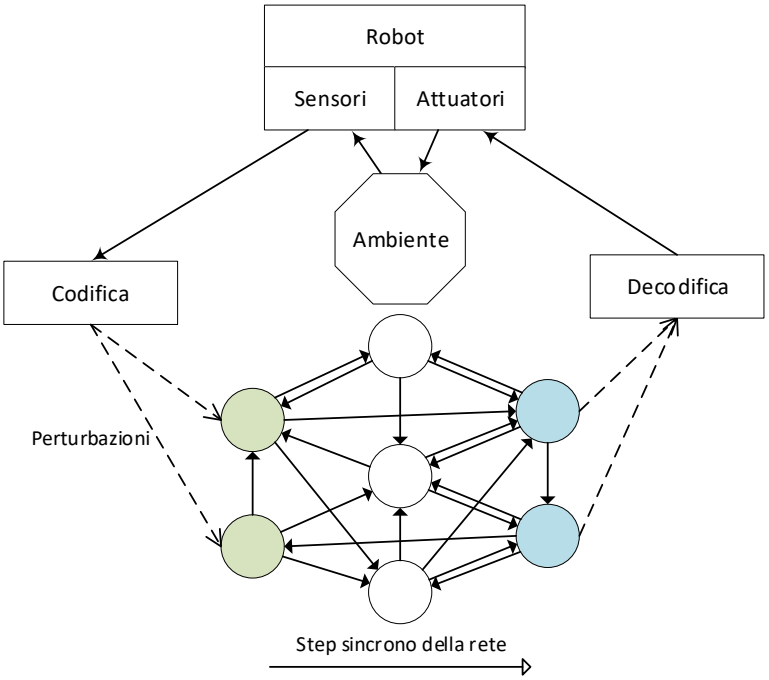


Figura 2.2: Schema di interazione tra ambiente, robot e rete booleana

Per ogni simulazione effettuata si è fissato Δt a 0.1 secondi, dunque, ogni robot avrà a disposizione 10 cicli di esecuzione al secondo per elaborare le informazioni e attuare la propria logica di controllo. Ad ogni ciclo di esecuzione un robot effettua le seguenti operazioni:

1. legge i valori dai 24 sensori di prossimità[2].

2. Converte le 24 letture numeriche in 8 valori booleani attraverso l'aggregazione in gruppi di 3, mantenendo i valori massimi, e utilizzando una soglia per determinare i valori booleani (fase di codifica).

[2]In alcune varianti verranno utilizzati anche i valori letti dai sensori di luminosità e da alcuni sensori virtuali. Le varianti sono descritte nella sottosezione Task.

3. Perturba la rete nei relativi nodi di input con i valori booleani ricavati in precedenza.

4. Esegue un passo sincrono sulla rete booleana.

5. Converte i valori booleani dei nodi di output in segnali di controllo per gli attuatori. Sono presenti due nodi di output che controllano i due motori (sinistra, destra). Il segnale 1 viene interpretato come piena potenza al motore e, viceversa, il segnale 0 indica che il motore deve rimanere spento. Ciò permette al robot una mobilità completa.

L'ambiente circostante chiude il ciclo sensomotorio del robot (*sensorimotor loop*). Le percezioni del robot perturbano lo stato della rete booleana, la quale produce stimoli per gli attuatori del robot, i quali, a loro volta lo fanno interagire con l'ambiente. Il comportamento che manifesta un robot è quindi influenzato pesantemente dalle dinamiche interne della rete booleana e dalla combinazione dei nodi di input e output scelti.

Con questa forma di controllo è facilmente riproducibile un'architettura reattiva in quanto un nodo di input potrebbe direttamente influenzare un nodo di output. Ad esempio, è semplice immaginare che un veicolo di Braitenberg possa essere implementato usando una semplice rete booleana con due nodi di input direttamente connessi a due nodi di output. Tuttavia, non è escluso che questa particolare architettura di controllo non possa manifestare comportamenti anche più complessi, nei quali, potrebbe essere richiesta la presenza di una memoria interna e di una complicata elaborazione dei segnali. Ad esempio, in un esperimento in cui è stata usata questa architettura di controllo è mostrato come un robot riesca a compiere un task di fototassi e di anti-fototassi con la stessa rete booleana, e come questa riesca a cambiare comportamento dopo la percezione di un segnale acustico [9]. Questo implica che l'informazione del segnale viene memorizzata all'interno della rete booleana la quale causa un cambiamento di dinamiche interne tale da manifestare un comportamento opposto. In questo progetto l'obiettivo è quello di individuare quale tipologia di rete booleana e quale forma di adattamento sia ottimale nei vari task progettati. Nella sottosezione Task sono stati definiti alcuni obiettivi in cui è necessaria la presenza di memoria e altri in cui sarebbe sufficiente anche il solo controllo reattivo. I robot per adattarsi possono cambiare combinazione di nodi di input/output o modificare la struttura interna della rete. Queste tecniche sono descritte in dettaglio nella sottosezione Tecniche di adattamento.

Topologia delle RBN usate

Ogni robot è dotato inizialmente di una rete booleana casuale (RBN) generata sulla base dei parametri N, K e p. Alcuni parametri della rete vengono fissati in tutte le simulazioni per diminuire lo spazio di configurazioni da testare, in particolare si è scelto di fissare N (numero di nodi della rete) a 100 e K (grado entrante) a 3. In questo modo agendo soltanto sul parametro p (*bias* di

generazione delle tavole di verità dei nodi) è possibile generare reti booleane caratterizzate da dinamiche interne ordinate, caotiche o critiche. Lo stato iniziale della rete booleana è scelto casualmente tra uno dei 2^{100} possibili. Altri vincoli applicati alla generazione della rete sono:

- gli archi devono collegare nodi distinti (no *self-loops*).
- Ogni nodo deve avere K nodi entranti distinti.
- Un nodo non può essere usato contemporaneamente come nodo di input e di output.

Al fine di evitare che l'effetto del parametro p incida direttamente sugli attuatori (*bias* di attivazione) si è deciso che ogni nodo di output è caratterizzato da una tavola di verità aggiuntiva che sovrascrive quella originaria. Questa nuova funzione sarà generata utilizzando sempre $p = 0.5$ in modo da non avere un *bias* sull'attivazione o disattivazione del nodo di output collegato all'attuatore. Quando e se il nodo di output tornerà ad essere un semplice nodo interno della rete, la funzione aggiuntiva verrà scartata lasciando spazio alla tavola di verità originaria.

2.3.3 Task

Il task da compiere viene espresso tramite una funzione obiettivo che i robot utilizzano durante l'adattamento. Ogni simulazione è suddivisa in 200 epoche dalla durata di $d = 80$ secondi. Al termine di ogni epoca un robot determina la bontà della propria configurazione sulla base del punteggio (*fitness*[3]) calcolato attraverso la funzione obiettivo. Il punteggio viene accumulato per ogni passo di simulazione e resettato all'inizio di ogni nuova epoca. Ogni robot cerca di massimizzare il punteggio ottenuto nelle varie epoche andando ad applicare la forma di adattamento prestabilita.

Task I

Questo task viene assegnato soltanto nel primo ambiente (figura 2.1) e consiste nel muoversi il più possibile in linea retta, minimizzando il numero di svolte, ed evitando allo stesso tempo gli ostacoli incontrati. Per ottenere questo risultato si è definita una funzione obiettivo così strutturata:

$$\frac{100}{E} \cdot \sum_{n=1}^{E} (1 - \theta(n)) \cdot (1 - \sqrt{|l(n) - r(n)|}) \cdot \frac{l(n) + r(n)}{2} \tag{2.1}$$

dove:

[3]Benché il termine *fitness* sia tipico degli algoritmi genetici, in questa tesi lo si usa con un'accezione più ampia. Con questo termine si fa riferimento al grado di adattamento che un robot è riuscito a raggiungere secondo una certa misura (funzione obiettivo).

- $E = \frac{d}{\Delta t} = 800$ è il numero di passi in un'epoca.
- n è il passo attuale.
- $\theta(n)$ è il valore di prossimità massimo letto dai sensori presenti sul robot al passo n. $\theta(n) \in [0, 1]$. Minore è $\theta(n)$ più si è lontani da eventuali ostacoli.
- $l(n)$ indica la potenza erogata al motore sinistro al passo n, $l(n) \in \{0, 1\}$. Con 0 il motore è spento.
- $r(n)$ indica la potenza erogata al motore destro al passo n, $r(n) \in \{0, 1\}$. Con 0 il motore è spento.

Al termine di un'epoca un robot può aver accumulato un punteggio nell'intervallo $[0, 100]$. Questa funzione premia i robot che riescono a muoversi con un moto rettilineo evitando gli ostacoli.

Task II

Questo task viene assegnato soltanto nel secondo ambiente nel quale sono presenti due regioni virtuali distinte che suddividono l'arena in due parti uguali. Il punteggio viene calcolato come nella formula 2.1 introducendo un fattore di penalità ϵ che assume valore 1 quando il robot si trova nella regione corretta, -1 altrimenti. Si considera come regione corretta quella in cui il robot viene posizionato inizialmente durante la fase di schieramento. In questo task i robot devono muoversi come nel caso precedente ma soltanto nella regione prestabilita, evitando di entrare in quella errata. La nuova funzione obiettivo diventa:

$$\frac{100}{E} \cdot \sum_{n=1}^{E} \epsilon(n) \cdot (1 - \theta(n)) \cdot (1 - \sqrt{|l(n) - r(n)|}) \cdot \frac{l(n) + r(n)}{2} \tag{2.2}$$

dove $\epsilon(n)$ assume valore 1 se il robot al passo n si trova sulla propria regione, -1 altrimenti. Al termine di un'epoca un robot può aver accumulato un punteggio nell'intervallo $[-100, 100]$. Al robot viene fornita l'informazione della regione corretta / errata su un nodo di input aggiuntivo. Oltre agli 8 nodi di input per i sensori di prossimità se ne aggiunge uno che indica al robot su quale regione si trova. Questa informazione viene estrapolata da un sensore virtuale di posizione e propagata durante la fase di perturbazione della rete. Al termine di ogni epoca la regione di riferimento viene resettata per consentire al robot di trovarsi sempre sulla regione corretta all'inizio di una nuova epoca.

Task III

Questo task viene assegnato nel terzo e quarto ambiente. Esso è un task di *foraging* dove i robot devono raccogliere un oggetto virtuale nella zona azzurra e portarlo nella zona verde (figura 2.1). Per fare ciò i robot sono dotati di un

gancio (anche questo virtuale) che è comandato da un segnale binario: raccogli, deposita. Inoltre, per aiutare i robot ad orientarsi è stata disposta una fonte luminosa sopra la zona verde, così da fornire un gradiente da seguire. Tutte queste informazioni sono propagate sulle reti booleane attraverso nodi di input aggiuntivi. L'informazione della regione in cui si trova un robot è propagata attraverso due segnali booleani (tre di quattro combinazioni sfruttate):

- **00** zona neutra
- **01** zona verde
- **10** zona azzurra

I segnali luminosi, come quelli di prossimità, sono propagati attraverso otto segnali booleani. In totale la rete del robot viene perturbata con $8+8+2 = 18$ valori booleani (prossimità, luminosità, zona) e vengono ricavati $2 + 1 = 3$ valori booleani per controllare gli attuatori (motori, gancio).

Viene usata la stessa funzione obiettivo del primo task per spingere i robot a muoversi ed esplorare l'arena, con la differenza di penalizzare ulteriormente i robot che collidono ($\theta(n)$ moltiplicato per 2).

$$\sum_{n=1}^{E} \frac{100}{E} \cdot (1 - 2\theta(n)) \cdot (1 - \sqrt{|l(n) - r(n)|}) \cdot \frac{l(n) + r(n)}{2} \tag{2.3}$$

Viene introdotto un premio che va a sommarsi al punteggio di *fitness* ogni qualvolta il robot esegue una di queste azioni:

- raccolta mentre il robot è sulla zona azzurra [+50]
- deposito mentre il robot è sulla zona verde [+50]
- deposito negli altri casi [−100]

L'azione di raccolta è valida solo se il robot si trova sulla regione corretta, mentre l'azione di deposito è possibile solo se il robot ha raccolto in precedenza un oggetto virtuale. In questo task non viene fornita al robot l'informazione di possesso di un oggetto, infatti, esso deve riuscire a mantenere in memoria lo stato in cui si trova. Un feedback potrebbe derivare dalla posizione del gancio, infatti, se fosse chiuso il robot potrebbe supporre di possedere un oggetto ma ciò non è garantito. Il task è stato progettato per obbligare il software di controllo a memorizzare un'informazione e, in base a questa, modulare il piano d'azione. Il sistema a premi dovrebbe spingere i robot a spostare quanti più oggetti possibili dalla zona verde a quella azzurra, facendosi guidare dal gradiente luminoso ed evitando gli ostacoli. Questo task assume una difficoltà differente in base all'arena in cui viene applicato: nella terza il compito è relativamente semplice in quanto gli ostacoli permettono facilmente ai robot di filtrare tra le due regioni d'interesse, invece, nella quarta arena gli ostacoli sono disposti in modo da intralciare maggiormente lo spostamento.

2.3.4 Tecniche di adattamento

Le tecniche di adattamento prevedono di modificare qualche caratteristica del software di controllo, basandosi sul punteggio di *fitness* ottenuto. Ogni robot mantiene la rete booleana (inclusi gli insiemi dei nodi di input e output) che gli ha permesso di ottenere il miglior punteggio e, partendo da questa, al termine di ogni epoca viene applicata una variazione per tentare di trovare una configurazione di migliore o uguale valore. La nuova configurazione viene testata nell'epoca successiva e confrontata sulla base del punteggio di *fitness* ottenuto. Nel caso in cui si ottiene un punteggio inferiore la nuova configurazione viene scartata. Invece, nel caso in cui si ottiene un nuovo punteggio massimo la configurazione attuale del robot diventa il riferimento da cui applicare nuovamente una delle tecniche di adattamento. Le tre forme di adattamento prevedono di:

1. lasciare immutata la struttura interna della rete, rimpiazzando casualmente un nodo di output e due nodi di input con altri nodi interni. Questa forma di adattamento sarà chiamata *ripartizione* da qui in avanti.

2. Modificare il collegamento dell'un percento degli archi in modo casuale e alterare l'un percento dei bit delle tavole di verità delle funzioni booleane. Questa forma di adattamento sarà chiamata *alterazione* da qui in avanti.

3. Applicare entrambe le forme di adattamento. Questa forma di adattamento sarà chiamata *ibrida* da qui in avanti.

Queste forme di adattamento sono state progettare per verificare questa ipotesi: partendo da RBN critiche (es $K = 3, p = 0.79$) non è necessario modificare la struttura interna della rete per ottenere una forma di controllo che massimizzi il punteggio, ma bensì, è sufficiente trovare la configurazione corretta di nodi di input e output che sfrutta appieno le capacità computazionali offerte dalla rete booleana. La prima forma di adattamento, infatti, non muta la struttura interna delle reti booleane e serve proprio per verificare la veridicità di questa ipotesi. La seconda forma di adattamento, invece, serve per poter paragonare le due tecniche e trarre alcune conclusioni. Quest'ultima può essere paragonata ad una tecnica di robotica evolutiva dove la popolazione è costituita da un singolo individuo e la mutazione è applicata sulla struttura interna delle reti booleane. La prima forma di adattamento, invece, è paragonabile all'applicazione della plasticità fenotipica. Quest'ultima è la capacità di un individuo di svilupparsi in differenti fenotipi in relazione a diverse condizioni ambientali [8]. In questo caso l'individuo (genotipo) è la rete booleana del robot.

Nel capitolo 3 si discute dell'effetto di queste tre tecniche basandosi sul punteggio di *fitness* ottenuto dai robot nelle varie simulazioni. Un ulteriore analisi è rivolta allo studio dell'effetto della mutazione sulle RBN introdotta dalla tecnica di *alterazione*. Infatti, applicando delle modifiche alla struttura interna della rete in modo non totalmente casuale (*bias* dato dalla funzione obiettivo) le reti risultanti non sono più *Random* [6]. Nulla vieta che una

RBN ordinata e sottoposta a mutazione diventi di fatto una rete booleana con caratteristiche critiche o caotiche.

2.4 Esperimenti

Ogni esperimento è costituito da un'insieme di configurazioni distinte. Una configurazione è l'insieme di paramenti (p, N, K, arena, task, numero di robot, ecc.) necessari ad avviare una simulazione. Le esecuzioni sono completamente deterministiche in modo da avere una riproducibilità dei risultati completa. Una simulazione avviata con una determinata configurazione produce sempre lo stesso risultato a parità di *random seed*. Per ogni configurazione, variando il *random seed*, vengono sempre create mille copie al fine di ottenere risultati statisticamente validi. In questo modo, per ogni esperimento si ottiene un insieme di risultati che permette di fare paragoni tra le diverse configurazioni con una buona confidenza. Il *random seed* incide sia sul posizionamento iniziale dei robot, sia sulla struttura e lo stato iniziale delle reti booleane. Inoltre, la funzione *random* viene usata anche durante l'applicazione delle tecniche di *alterazione* e *ripartizione*.

L'esperimento principale di questo progetto ha come obiettivo quello di valutare l'effetto del parametro p, le tecniche di adattamento e le applicazioni di esse nei vari task descritti in precedenza. Queste valutazioni saranno quasi sempre sulla base dei punteggi di *fitness* prodotti dai robot. Di seguito è riportata una tabella con i valori dei parametri sperimentati:

p	Adattamento	Arena e Task
0.1	*ripartizione*	Arena I, Task I
0.5	*alterazione*	Arena II, Task I
0.79	*ibrida*	Arena III, Task III
		Arena IV, Task III

Tabella 2.1: Tabella dei parametri sperimentati.

Le combinazioni di questi paramenti producono un totale di $3 \cdot 3 \cdot 4 = 36$ configurazioni distinte. Tutti gli altri parametri della simulazione sono fissati a valori precisi: $N = 100$, $K = 3$, *no self loop*, $\Delta t = 0.1$, $d = 80$s, Robots=10 [4]. La scelta di fissare i parametri ad un determinato valore è stata necessaria per ristringere il campo di ricerca. Tuttavia, sono stati effettuati alcuni test per determinare valori ragionevoli, ma questi non vengono descritti per evitare una discussione che si discosta dall'argomento della tesi. I risultati dell'esperimento riportato in tabella 2.1 fa da guida per la definizione di ulteriori analisi che saranno presentate nei capitoli successivi. Il prossimo capitolo

[4]Per una lista completa dei parametri si può far riferimento all'appendice A

espone i risultati prodotti da queste 36 diverse configurazioni al fine di raccogliere prove sull'effetto della forma di adattamento combinato alla tipologia di rete booleana utilizzata.

Capitolo 3

Risultati

In questo capitolo sono mostrati i risultati ricavati dalla prima serie di esperimenti. Si discute della differenza di prestazioni ottenute dalle tre tecniche di adattamento, e dalla variazione del parametro p che incide sulla generazione delle RBN. Infine, si valuta l'impatto della mutazione sulla struttura interna delle reti booleane.

3.1 Confronto tra tecniche di adattamento

In questa sezione si mettono a confronto le tre tecniche di adattamento usate: *ripartizione*, *alterazione* e *ibrida*. Di seguito vengono riproposte brevemente le differenze:

Ripartizione al termine di ogni epoca il robot rimpiazza due nodi di input e uno di output in modo casuale. Ciò fa sì che la rete verrà perturbata dai valori dei sensori in regioni diverse, e l'output per gli attuatori sarà ricavato da una combinazione di nodi differente. La struttura interna della rete booleana rimane invariata.

Alterazione al termine di ogni epoca il robot altera la struttura interna della rete booleana nel seguente modo: l'un percento delle voci delle tavole di verità (8 bit) vengono negate, e l'un percento delle connessioni tra nodi (3 archi) vengono ridistribuite. Sia le voci che gli archi vengono scelti casualmente. La struttura interna della rete quindi muta, mentre l'insieme di nodi di input e output rimane invariato. In seguito, quando si usa il termine "mutazione", si richiama proprio il fatto che questa tecnica di adattamento, e quella *ibrida*, mutano la struttura interna della rete booleana.

Ibrida in questo caso si applicano entrambe le tecniche.

Per l'analisi dei risultati sono utilizzate due tipologie di grafici:

Andamento medio di fitness In questa tipologia di grafico ogni linea rappresenta l'andamento medio del punteggio di *fitness* nelle 200 epoche di una simulazione. L'andamento viene calcolato usando la curva di *fitness* di ogni robot per poi farne una media. La curva di *fitness* di un robot è composta da 200 punti (uno per epoca) ed ogni punto rappresenta il punteggio di *fitness* massimo raggiunto da quel robot in tutte le epoche precedenti. L'andamento è sempre crescente ed è un buon compromesso per rappresentare la capacità di adattamento dei robot.

Box-plot di fitness massima In questa tipologia di grafico ogni box-plot è composto dal massimo punteggio di *fitness* di ciascun robot. Come *fitness* massima s'intende il maggior punteggio che un robot sia riuscito a raggiungere in una delle 200 epoche a disposizione.

I grafici seguenti mostrano i risultati aggregati per tecnica di adattamento e suddivisi per le quattro distinte arene. La variazione del parametro p viene collassata e il suo effetto sarà discusso nella sezione successiva. In figura 3.1 è mostrato l'andamento medio del punteggio di *fitness* dei robot. In tutte e quattro le arene, e dunque in tutti i task assegnati, i grafici mostrano che l'utilizzo della sola *alterazione* non è una forma di adattamento efficace in questo contesto. Alcune cause potrebbero essere la durata limitata delle simulazioni, o le percentuali di modifica troppo basse. Inoltre, è bene sottolineare che non trattandosi di un algoritmo di genetica evolutiva non vi è condivisione d'informazione tra la popolazione della simulazione, infatti, ognuno dei 10 robot cerca di adattarsi con le sole capacità computazionali a sua disposizione. La tecnica di *ripartizione*, invece, sembra particolarmente efficace, e utilizzandola contemporaneamente alla tecnica di *alterazione* (forma *ibrida*) sembra avere un ulteriore vantaggio in termini di *fitness*. Questo risultato rientra nelle aspettative in quanto la tecnica di *ripartizione* offre un'adattabilità molto elevata se la rete sottostante offre capacità computazionali utili al task da compiere. Usare una differente combinazione di nodi di input o output comporta un cambiamento molto più radicale rispetto a mutare qualche funzione booleana o qualche connessione. Tuttavia, se la rete del robot non offre dinamiche interne utili, la sola tecnica di *ripartizione* non può portare all'adattamento del robot. In questi casi la forma *ibrida* può essere molto efficace in quanto permette di trovare una combinazione di nodi di input e output adeguata, e di raffinare le dinamiche interne della rete attraverso l'alterazione delle funzioni booleane o dei collegamenti tra nodi.

Gli andamenti, in tutti i grafici, tendono ad aumentare anche al raggiungimento dell'ultima epoca e ciò significa che vi sarebbe ancora un margine di miglioramento. Tuttavia, lo studio non si concentra sull'andamento asintotico del punteggio di *fitness*, ma si focalizza sul determinare quale tipologia di rete booleana (combinata con una tecnica di adattamento) è più predisposta ad adattarsi velocemente al task, alla fisicità del robot ed all'ambiente in cui si

trova. In questo particolare caso la tecnica *ibrida* è quella che mediamente permette ai robot di adattarsi con più successo.

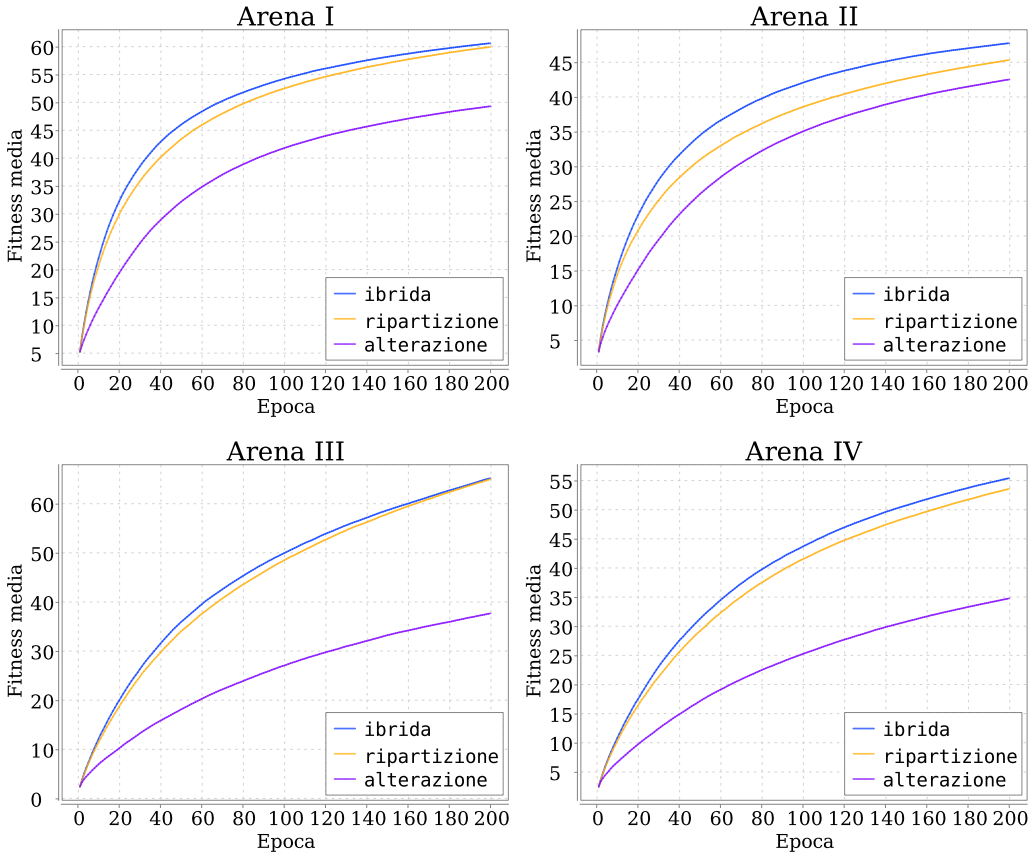


Figura 3.1: Andamento medio del punteggio di *fitness* rispetto alla tecnica di adattamento applicata. I risultati sono suddivisi per arena.

Anche i box-plot in figura 3.2 sembrano sostenere queste affermazioni. Tuttavia, l'utilizzo della sola *alterazione* non esclude il raggiungimento di risultati ottimi in alcuni casi. Invece, nella terza e quarta arena, dove il task da svolgere è più complicato, la tecnica di *alterazione* comporta un marcato svantaggio rispetto alle altre due forme di adattamento.

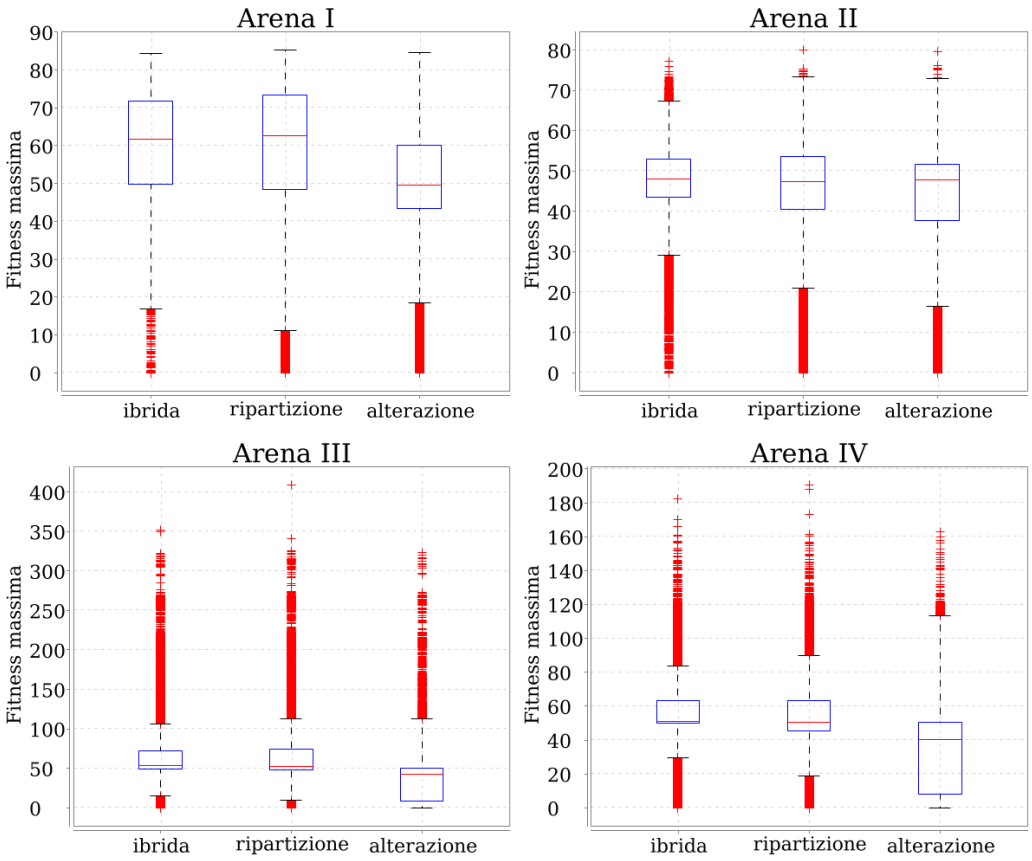


Figura 3.2: Box-plot generati dal punteggio di *fitness* massimo di ogni singolo robot rispetto alla tecnica di adattamento utilizzata.

3.2 Confronto dei parametri di generazione delle RBN

In questa sezione viene analizzato l'effetto del parametro p, mettendolo in relazione alla tecnica di adattamento utilizzata, sulla base dei punteggi di *fitness* ottenuti. Nella figura 3.4 è mostrato l'andamento del punteggio di *fitness* mettendo in risalto la relazione tra tipologia di rete booleana e tecnica di adattamento utilizzata[1]. È evidente che le reti booleane generate con $p = 0.5$ ottengono un punteggio di *fitness* medio inferiore indipendentemente dalla tecnica di adattamento utilizzata. Si può anche notare che la configurazione $\{p = 0.79$, *ripartizione*$\}$ ottiene quasi sempre i risultati migliori, seguita dalla configurazione $\{p = 0.1$, *ibrida*$\}$. Questo confermerebbe l'ipotesi iniziale, infatti, le RBN critiche hanno già tutte le caratteristiche computazionali per svolgere il task richiesto, e dunque, la tecnica di *ripartizione* è sufficiente per ottenere buoni risultati. Invece, le reti inizialmente ordinate producono

[1]In questa sezione la tecnica di adattamento *alterazione* non viene più inclusa nei grafici per facilitare l'esposizione.

risultati migliori usando la forma *ibrida*, causando quindi la mutazione della struttura interna della rete. Nella prossima sezione si studia l'effetto della mutazione sulle reti booleane, in particolare, ci si concentra sulla configurazione $\{p = 0.1,$ *ibrida*$\}$ in quanto riesce a raggiungere ottimi risultati. Un'ipotesi è che le reti sottoposte a mutazione diventano di fatto reti booleane con caratteristiche critiche.

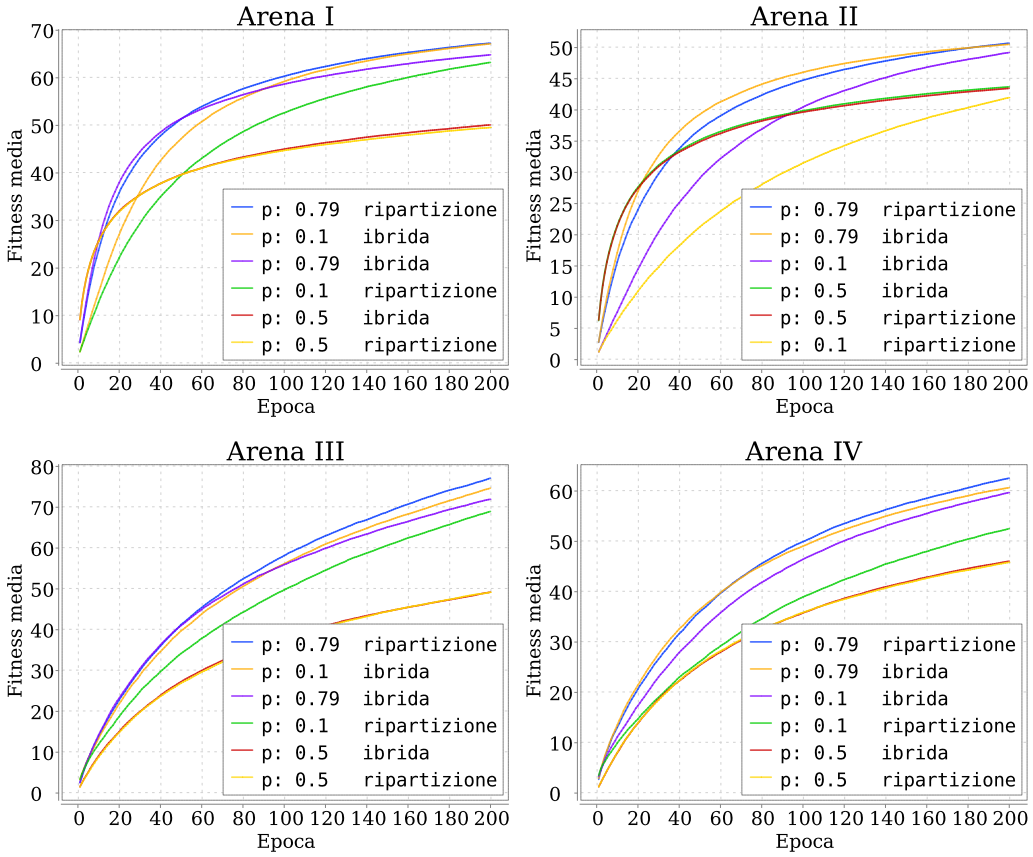


Figura 3.3: Andamento medio della *fitness* rispetto al parametro p e alle tecniche di adattamento *ripartizione* e *ibrida*. I grafici sono suddivisi per arena.

Nel grafico 3.4 sono mostrati i punteggi di *fitness* massimi dei singoli robot suddivisi per categoria. Anche questi dati confermano che le reti booleane generate con $p = 0.5$ ottengono risultati mediamente inferiori. Infatti, i robot migliori di questa categoria non riescono a raggiungere livelli elevati di *fitness* nei vari task. Nella quarta arena gli *outlier* con punteggio migliore provengono proprio dalla configurazione $\{p = 0.79,$ *ripartizione*$\}$. Questo dimostra che per alcuni casi fortuiti le RBN critiche offrono tutte le caratteristiche computazionali necessarie al robot (evitamento ostacoli, fototassi, raccolta, deposito) senza dover mutare la propria struttura interna. Ciò non accade mai per le altre tipologie di reti booleane (considerando soltanto la tecnica di *ripartizione*).

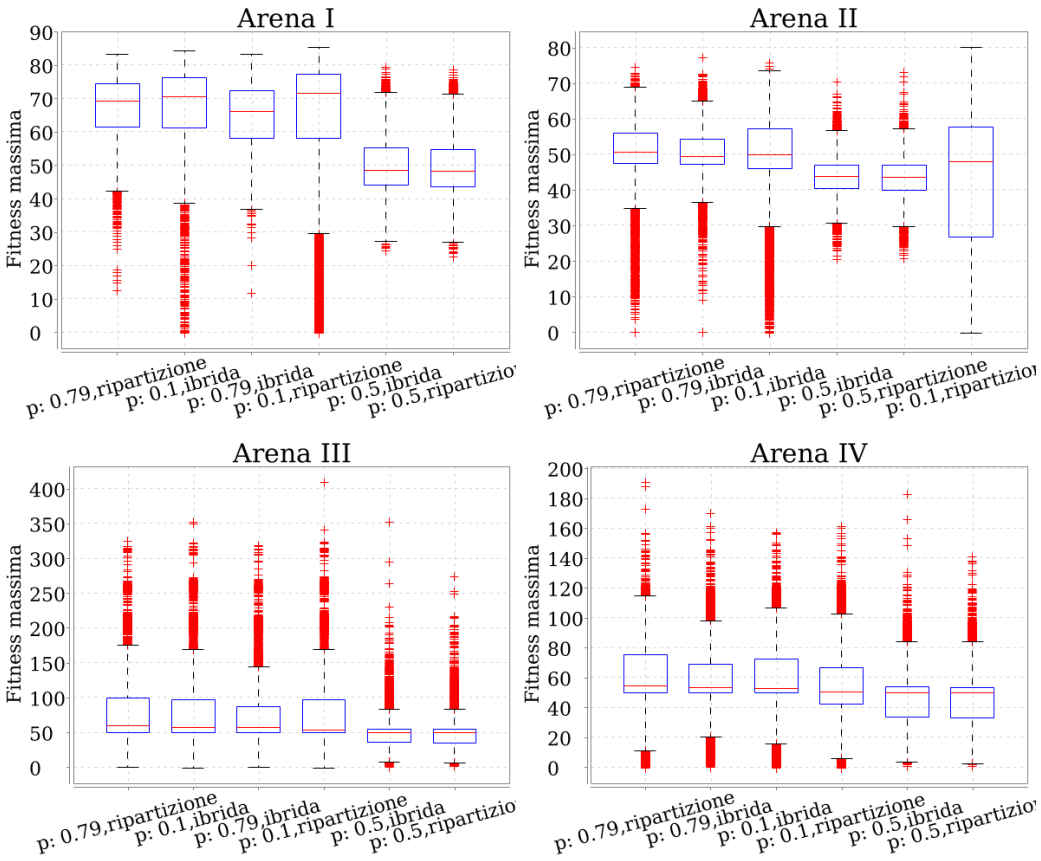


Figura 3.4: Box-plot generati dal punteggio di *fitness* massimo di ogni singolo robot. I grafici sono suddivisi per arena e viene preso in esame il parametro p e le tecniche di adattamento *ripartizione* e *ibrida*.

Un'altra possibile analisi è quella di tenere in considerazione solo la popolazione d'*elite* che è stata generata dalle varie configurazioni. Per ogni arena sono stati presi i 100 migliori robot (su 90000) sulla base del punteggio di *fitness* massimo. In seguito, si calcola la suddivisione della popolazione in base alla configurazione di appartenenza. I risultati sono suddivisi per arena in quanto essa determina la difficoltà complessiva del task da compiere, e quindi lo sforzo di adattamento dei robot. La tabella 3.1 mostra per ogni possibile combinazione di parametri la percentuale di popolazione d'*elite* generata. Il 43% dei robot d'*elite* nella prima arena sono costituiti da reti booleane generate con $p = 0.1$. Questo risultato è contrario all'ipotesi iniziale in cui le reti caratterizzate da dinamiche interne critiche dovrebbero dare al robot un vantaggio nell'adattamento. Tuttavia, è possibile notare come la percentuale di robot d'*elite* appartenenti al valore $p = 0.79$ aumenti al crescere della difficoltà del task. Nella quarta arena i robot d'*elite* inizializzati con una rete booleana critica sono addirittura in numero superiore rispetto a quelli con reti ordinate. Questo fenomeno suggerisce che con task semplici le reti ordinate ottengono prestazioni migliori, mentre, in task complessi dove i robot vincenti

devono manifestare comportamenti compositi sembrano avere un vantaggio le reti booleane critiche. Si può anche concludere che quando si considera la popolazione d'*elite*, la tecnica di *ripartizione* è preferibile rispetto a quella *ibrida*. In uno studio futuro si potrebbe aumentare ulteriormente la difficoltà del task per confermare o smentire queste affermazioni.

Arena I

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.43	0.04	0.00
ibrida	0.27	0.04	0.00
alterazione	0.19	0.03	0.00
Totale	0.89	0.11	0.00

Arena II

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.42	0.12	0.02
ibrida	0.12	0.10	0.00
alterazione	0.15	0.07	0.00
Totale	0.69	0.29	0.02

Arena III

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.30	0.19	0.01
ibrida	0.21	0.12	0.02
alterazione	0.09	0.06	0.00
Totale	0.60	0.37	0.03

Arena IV

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.17	0.21	0.04
ibrida	0.19	0.18	0.04
alterazione	0.08	0.06	0.03
Totale	0.44	0.45	0.11

Tabella 3.1: Le tabelle indicano le percentuali di popolazione d'*elite* generate dalle configurazioni {parametro p, tecnica di adattamento} per le quattro differenti arene.

3.3 Effetto della mutazione

In questa sezione si studia l'effetto della mutazione sulla struttura e sulle dinamiche interne delle reti booleane. Tale mutazione è introdotta dall'utilizzo delle tecniche di adattamento *alterazione* e *ibrida*. Durante la fase di adattamento le RBN sottoposte a mutazione non possono più essere considerate *Random* per via del *bias* di selezione indotto dalle tecniche di adattamento. Utilizzando il grafico di Derrida si studia la propagazione media delle perturbazioni nelle varie reti booleane e si confronta questo dato con quello ricavato dalle reti non sottoposte a mutazioni. Infine, si analizza il numero di stati distinti visitati dalle varie tipologie di reti booleane.

3.3.1 Valore di Derrida

Il valore di Derrida [4] può essere calcolato per una qualsiasi rete booleana in un determinato stato. Il procedimento è il seguente: si crea una copia dello stato della rete booleana e si nega il valore di una variabile di un nodo scelto

casualmente, successivamente, si esegue un passo sincrono per entrambi gli stati della rete e si misura la distanza di Hamming tra i due risultati. La distanza di Hamming è applicata sulle sequenze dei valori booleani dei nodi dei due stati distinti. L'algoritmo è schematizzato in figura 3.5. Questo procedimento viene ripetuto molte volte, perturbando nodi differenti, per determinare un valore medio. Il risultato misura il livello di propagazione medio di una perturbazione. Nelle reti caotiche questo valore è tipicamente maggiore di 1, nelle reti ordinate è inferiore a 1 e nelle reti critiche è 1.

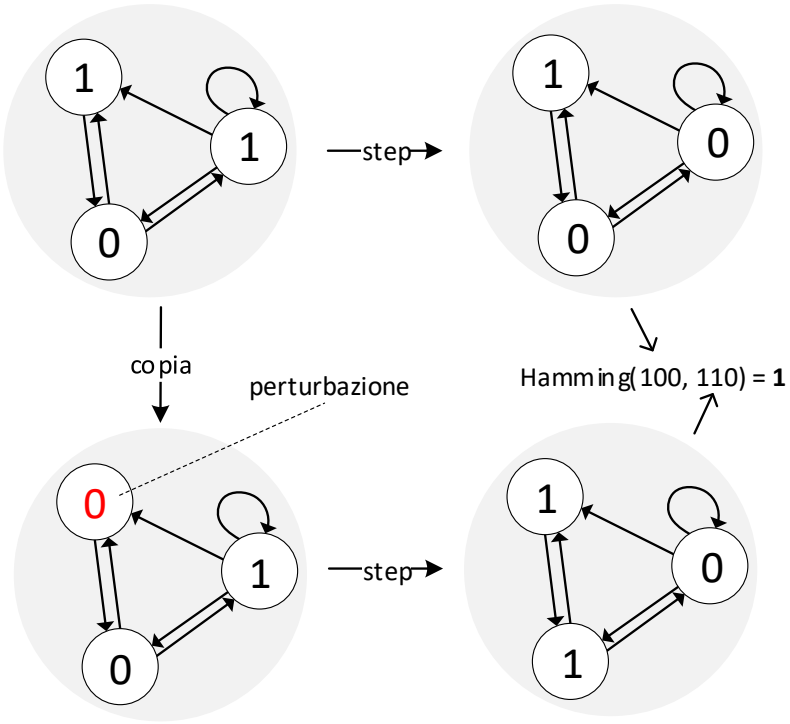


Figura 3.5: Schema del procedimento per calcolare il valore di Derrida. Si parte dalla rete booleana raffigurata in alto a sinistra. Si clona la rete e si perturba un nodo della copia. Successivamente, per entrambe le reti si effettua un passo sincrono. Infine, si calcola la distanza di Hamming tra gli stati risultanti.

Viene utilizzata questa misurazione in quanto altri strumenti, come *annealed approximation* [5], non sono appropriati quando si parla di RBN mutate attraverso un procedimento evolutivo [6]. In particolare, le reti sottoposte ad un'evoluzione non possono essere più classificate in ordinate, critiche o caotiche in quanto, esse possono mostrate caratteristiche di tutte e tre le tipologie [12, 15]. Tuttavia, i grafici generati dal valore di Derrida rimangono un buono strumento per analizzare le dinamiche interne delle reti booleane in questione [6].

3.3.2 Grafico di Derrida

In questa particolare analisi si usa la misura di Derrida per generare grafici a dispersione atti a mettere in relazione il punteggio di *fitness* al valore medio di Derrida. Per ogni robot il valore di Derrida viene calcolato con il seguente procedimento:

- s'individua l'epoca in cui il robot ha ottenuto il miglior punteggio;
- si estrapola lo stato della rete per tutti gli 800 passi dell'epoca;
- si calcola il valore di Derrida per ognuno degli 800 stati e su tutte le perturbazioni possibili ($N = 100$). Viene calcolato un valore medio per tutte le misurazioni appartenenti al robot;
- infine, viene usato il valore medio ottenuto per generare un punto del grafico a dispersione.

Il valore così elaborato rappresenta il livello di propagazione medio di una perturbazione soltanto sugli stati in cui il robot è transitato durante l'epoca in questione. Nella figura 3.6 sono mostrati i grafici di Derrida ricavati dalle simulazioni svolte nelle quattro arene, con $p \in \{0.1, 0.5, 0.79\}$ e con tecniche di adattamento *ripartizione* (grafici di sinistra) e *ibrida* (grafici di destra). Nei casi in cui viene applicata la sola tecnica di *ripartizione* è possibile notare dei *cluster* ben suddivisi in relazione al valore di Derrida ed al parametro p utilizzato per generare le RBN. Nel momento in cui s'introduce anche la mutazione (tecnica di adattamento *ibrida*) le regioni iniziano a sovrapporsi. In particolare, è possibile notare che le zone verdi e rosse ($p = 0.79, p = 0.1$) si spostano verso destra, ottenendo un valore di Derrida mediamente più alto. L'effetto è particolarmente marcato per le reti inizialmente ordinate.

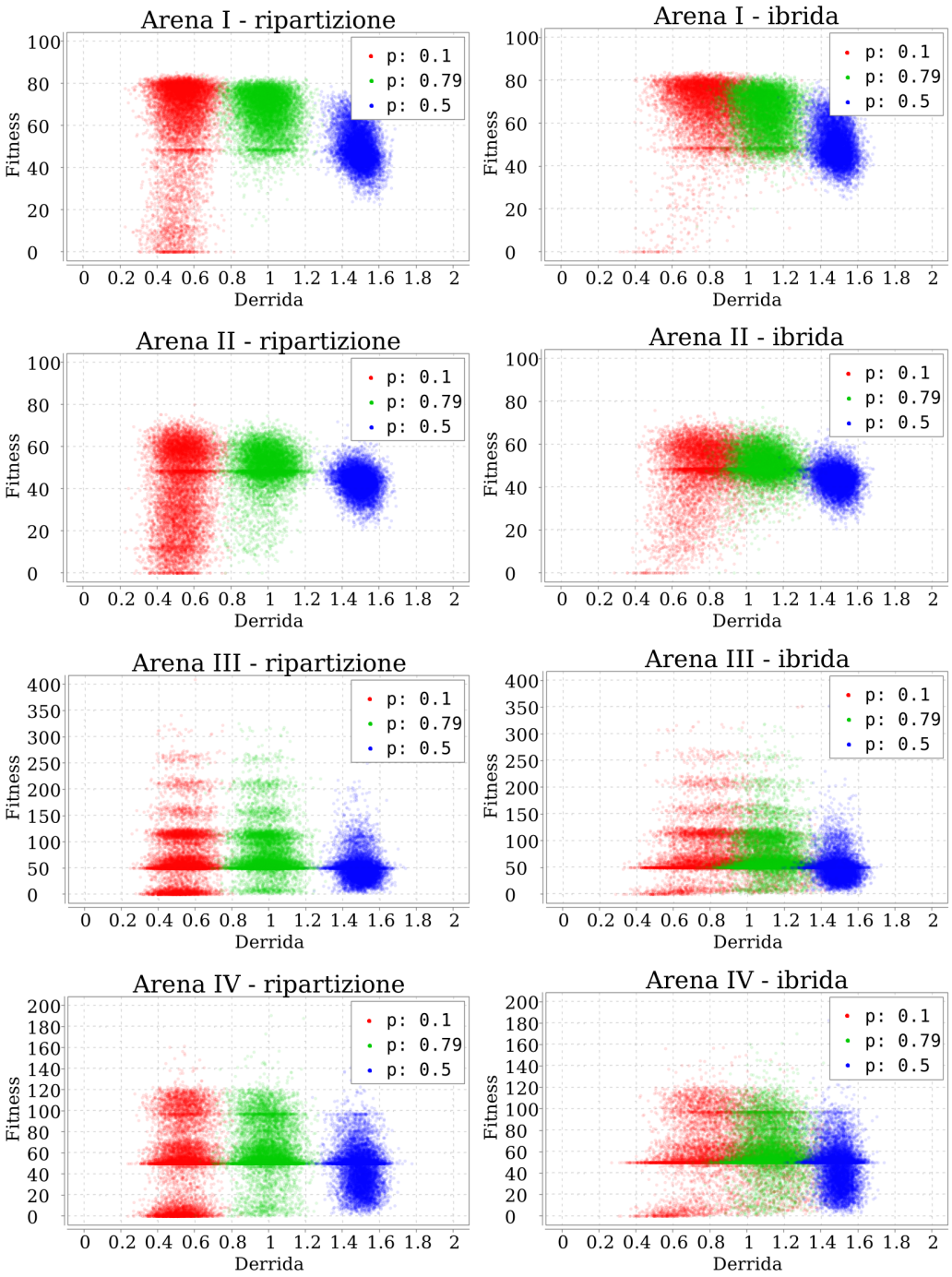


Figura 3.6: I grafici mostrano la relazione tra punteggio di *fitness* e valore di Derrida. Ogni punto è il risultato della miglior epoca di un robot. I grafici nella colonna di sinistra sono il risultato dell'applicazione della tecnica di adattamento *ripartizione*, in quelli di destra viene usata la tecnica *ibrida*.

Nelle tabelle 3.2 sono riportati i valori medi delle misure di Derrida delle tre regioni. Per quanto riguarda le reti originariamente ordinate, in tutte quattro le arene, l'uso della tecnica *ibrida* causa uno spostamento del valore medio di

Derrida verso l'1. Questo dato è a supporto dell'ipotesi iniziale in cui le reti generate con $p = 0.1$ e sottoposte a mutazione ottengono buoni risultati perché si avvicinano a dinamiche interne simili a quelle delle RBN critiche. Tuttavia, la sola misura di Derrida non è sufficiente per dimostrare che si tratta di reti critiche. Le reti caotiche, invece, non sembrano essere influenzate dalla mutazione e non sembrano ottenere buoni risultati in generale. Sicuramente questa tipologia di rete booleana non offre alcun vantaggio in termini di *fitness* rispetto a reti ordinate o critiche.

Arena I

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.54	0.99	1.49
ibrida	0.83	1.11	1.48
alterazione	0.92	1.16	1.47

Arena II

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.54	1.00	1.49
ibrida	0.82	1.11	1.49
alterazione	0.92	1.16	1.48

Arena III

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.54	0.99	1.49
ibrida	0.88	1.12	1.49
alterazione	0.97	1.17	1.49

Arena IV

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	0.54	0.99	1.49
ibrida	0.90	1.14	1.49
alterazione	0.98	1.18	1.49

Tabella 3.2: Le tabelle riportano i valori medi della misura di Derrida dei *cluster* illustrati nei grafici in figura 3.6. È evidente la tendenza delle reti generate con $p = 0.1$ ad avvicinarsi al valore 1 quando si utilizza la tecnica *ibrida* o *alterazione*.

3.3.3 Spazio degli stati

Un'ulteriore prospettiva è quella di contare gli stati visitati dalle reti booleane dei robot nelle loro miglior epoche. Per ogni robot si ripercorrono i passi dell'epoca in cui ha ottenuto il punteggio di *fitness* maggiore, contando il numero di stati distinti su cui la propria rete booleana è transitata. In generale una rete booleana ordinata transita su un numero di stati inferiore rispetto ad una rete critica; entrambe incorrono in meno stati di una rete caotica. Questo perché una rete ordinata che viene perturbata in pochi nodi ricade con probabilità maggiore sullo stesso attrattore, e comunque è composta da attrattori di lunghezza minima. Una rete caotica, invece, è caratterizzata da attrattori di lunghezza molto maggiore, quindi, indipendentemente dalle perturbazioni incorre in un numero maggiore di stati distinti. Dai dati raccolti e mostrati in figura 3.7 e nella tabella 3.3 si può osservare che questa considerazione è valida anche con l'introduzione della mutazione. Tuttavia, è evidente che le reti ordinate sottoposte a mutazione hanno un aumento del numero di stati visitati molto maggiore rispetto all'aumento ottenuto nelle reti critiche sottoposte

a mutazione. Di fatto, anche questa analisi sembra indicare una somiglianza tra le configurazioni {*p: 0.1, ibrida*} e {*p: 0.79, ripartizione*}. Queste configurazioni sono considerate importanti perché generalmente sono quelle che ottengono i migliori risultati. Inoltre, è particolarmente sorprendente la somiglianza di valori nella terza e quarta arena. Sembra che le reti ordinate per avere successo in task complicati mutino in direzione delle reti critiche. Questo dato fa da ulteriore supporto all'ipotesi iniziale.

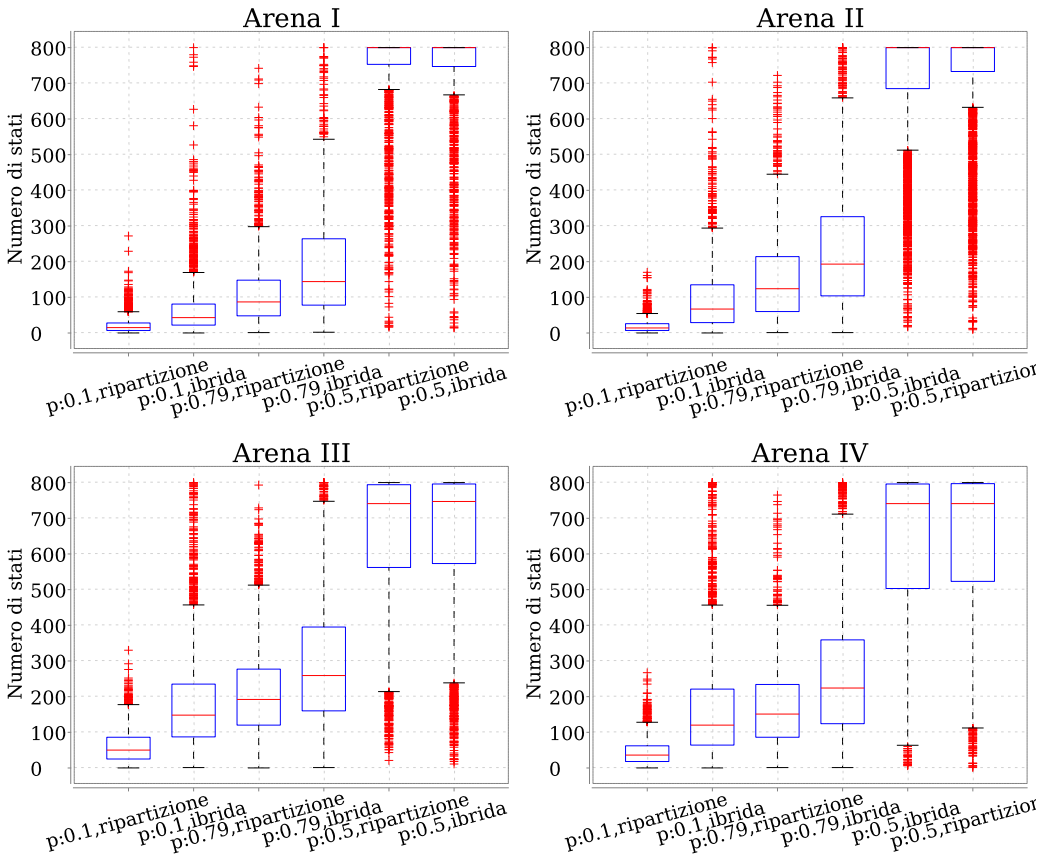


Figura 3.7: Box-plot generati dal numero di stati distinti in cui la rete booleana di un robot è transitata nella miglior epoca.

Arena I

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	27	158	715
ibrida	95	239	704
alterazione	132	232	584

Arena II

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	30	135	691
ibrida	89	205	680
alterazione	115	194	565

Arena III

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	48	178	567
ibrida	150	243	570
alterazione	193	245	533

Arena IV

	p		
Adattamento	***0.1***	***0.79***	***0.5***
ripartizione	38	139	478
ibrida	131	196	477
alterazione	169	215	456

Tabella 3.3: Nelle tabelle sono riportati i valori medi del numero di stati distinti visitati dalla rete booleana di un robot nella sua miglior epoca.

3.4 Conclusioni

Questo esperimento ha permesso di studiare gli effetti delle tecniche di adattamento e del valore del parametro p sulla base dei punteggi di *fitness* ottenuti in svariati task. Dai dati raccolti si può concludere che:

- i risultati migliori, in termini di *fitness*, derivano tipicamente dall'utilizzo di reti ordinate e critiche, escludendo in ogni situazione le reti caotiche.
- In task semplici, sia le reti booleane ordinate che quelle critiche ottengono ottimi risultati, prediligendo tuttavia reti di natura ordinata.
- All'aumentare della difficoltà del task le reti critiche ottengono un vantaggio, in termini di *fitness*, rispetto alle reti ordinate.
- Le reti ordinate sottoposte a mutazione acquisiscono alcune caratteristiche delle reti critiche. Questo effetto è particolarmente marcato nei task di difficoltà maggiore.

Questi risultati fanno da ulteriore prova alla congettura *"i sistemi viventi sono complessi"* e in generale alla correttezza dell'ipotesi iniziale. Tuttavia, non vi è una netta distinzione di prestazioni tra reti booleane critiche e ordinate, lasciando spazio ad ulteriori esperimenti futuri. Nel prossimo capitolo le varie reti booleane vengono sottoposte ad alcune tipologie di perturbazioni per determinare quali siano quelle più robuste.

Capitolo 4

Robustezza alle perturbazioni e ai guasti

In questo capitolo si analizza la robustezza alle perturbazioni ed ai guasti delle reti booleane utilizzate. Nella prima sezione viene introdotta una perturbazione continua sulle reti durante la fase di adattamento dei robot. La perturbazione può essere paragonata ad una radiazione di fondo che introduce rumore negli stati dei nodi. Nella seconda sezione vengono clonati e schierati alcuni robot d'*elite* ricavati dalle simulazioni del capitolo 3. In questi robot gli stati delle reti booleane vengono inizializzati ad un valore casuale per determinare se durante l'esecuzione essi riescano a tornare in un regime di funzionamento ottimale. Nell'ultima sezione s'introducono malfunzionamenti all'interno delle reti booleane e si studiano gli effetti sulle prestazioni. In particolare, si studia l'effetto dell'eliminazione di archi.

4.1 Perturbazioni continue

Con perturbazione s'intende una semplice inversione di stato di un nodo della rete. Se si considera $s(t)$ come stato attuale della rete booleana, una perturbazione avviene tra $s(t)$ e $s(t+1)$ e nega la variabile booleana di un nodo casuale. Viene definito un parametro ν che indica la frequenza di perturbazioni al secondo. Ad esempio, con $\nu = 50$ un robot viene sottoposto in media a 50 perturbazioni al secondo. La provocazione di questi eventi è determinata da una distribuzione esponenziale, quindi, dato l'instante attuale t_0 per calcolare l'instante della prossima perturbazione t_e si usa la seguente legge:

$$t_e = \frac{-ln(1-x)}{\nu} + t_0 \tag{4.1}$$

x è un valore estratto a sorte tra $]0,1[$ con distribuzione uniforme. Quando il tempo supera t_e, viene scelto un nodo casuale della rete e si inverte lo stato. In seguito, si calcola un nuovo t_e. Questo test è utile per capire quale tipologia

di rete booleana sia più robusta rispetto ad un rumore di fondo o ad un errore di calcolo durante l'esecuzione. Questo esperimento, a differenza di quello nella prossima sezione, prevede di applicare le perturbazioni durante la fase di adattamento e non di testare reti booleane già evolute. Lo scopo è quello di capire quale tecnica di adattamento e tipologia di rete booleana riesca a raggiungere livelli di *fitness* più elevati in queste determinate condizioni. Le reti caotiche non sono state testate per via della loro natura. Infatti, si presuppone che ogni perturbazione si propagherebbe molto velocemente nella rete, affliggendo le prestazioni in maniera più radicale rispetto alle altre tipologie di reti. Inoltre, come mostrato nei capitoli precedenti, le reti generate da $p = 0.5$ (inizialmente caotiche) ottengono sempre risultati inferiori e quindi si è scelto di escluderle anche per risparmiare tempo di simulazione.

In questo esperimento vengono testati i seguenti parametri nelle quattro arene: $p \in \{0.1, 0.79\}$, $\nu \in \{0, 1, 10, 100, 1000\}$ e tecniche di adattamento *ripartizione* e *ibrida*. Per ogni combinazione di parametri sono state effettuate mille simulazioni al fine di ottenere risultati statisticamente validi. Per questo esperimento sono state svolte un totale di 32000 simulazioni con dieci robot ciascuna. I restanti parametri della simulazione sono rimasti invariati rispetto ai casi precedenti e possono essere consultati nell'appendice A.

Come primo risultato si mostra l'effetto del solo parametro ν, collassando gli altri. Con questa prima indagine si vuole determinare un valore di ν ragionevole per approfondire questa analisi. In figura 4.1 è mostrato l'andamento medio di *fitness*[1] e i punteggi di *fitness* migliori di ogni robot sulla base del parametro ν. È evidente che all'aumentare di ν l'andamento medio di *fitness* subisce un calo. Con $\nu = 1$ si ottengono risultati simili ai casi in cui non si ha rumore ($\nu = 0$), con $\nu = 1000$ l'effetto della perturbazione è troppo significativo, infatti, dal box-plot si può vedere che anche tra gli *outlier* non vi è alcun valore elevato. Pertanto, si è deciso di studiare più in dettaglio l'effetto di $\nu = 100$ in quanto questo valore offre un buon livello di rumore, permettendo comunque ai robot di adattarsi con un buon successo. Ogni robot subisce in media dieci perturbazioni ad ogni passo di simulazione ($\nu \cdot \Delta t = 10$).

[1]Per questa analisi si utilizzano gli stessi grafici descritti nel capitolo Risultati

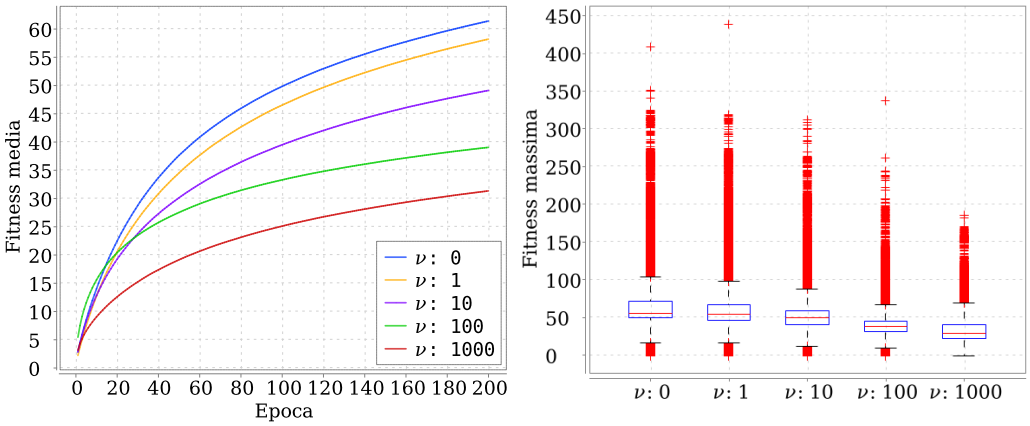


Figura 4.1: Andamento medio di *fitness* e *fitness* massima dei robot sottoposti a diverse frequenze di perturbazione. ν è il numero di perturbazioni al secondo. I dati sono aggregati sia per tipologia di arena, valore di p e tecnica di adattamento.

Dai risultati raccolti e mostrati in figura 4.2 emerge che le reti generate con $p = 0.79$ ottengono i risultati migliori anche se la differenza di prestazioni non è troppo marcata in alcuni casi. La differenza più evidente è che molti robot dotati di reti booleane generate con $p = 0.1$ ottengono punteggi di *fitness* molto bassi, causando un abbassamento dell'andamento medio di *fitness*. Questi risultati sembrano essere conformi all'ipotesi di criticità, la quale afferma che i sistemi critici mostrano comportamenti caratterizzati da un equilibrio ottimale tra robustezza e adattabilità. Tuttavia, la differenza nei risultati è piuttosto limitata e occorrerebbe uno studio statistico più approfondito per affermare la significatività delle differenze osservate.

Con questi dati e con quelli raccolti nel capitolo 3 è possibile asserire che in linea generale è sempre preferibile partire da reti booleane critiche, e da queste applicare una delle forme di adattamento. Generalmente queste reti riescono ad ottenere ottimi risultati anche senza l'ausilio della mutazione, a differenza di quelle ordinate o caotiche.

È particolarmente sorprendente che con $\nu = 100$ le reti in questione riescano a adattarsi con ottimi risultati. In particolare, nella terza e quarta arena alcuni robot riescono persino a raggiungere livelli elevati di *fitness*. Considerando che in media subiscono dieci perturbazioni ad ogni passo di simulazione questo risultato è ottimo da un punto di vista della robustezza di questa architettura di controllo. Ciò significa che in un'epoca alcuni robot riescono comunque ad effettuare delle consegne con successo seppur sottoposti a forti perturbazioni esterne.

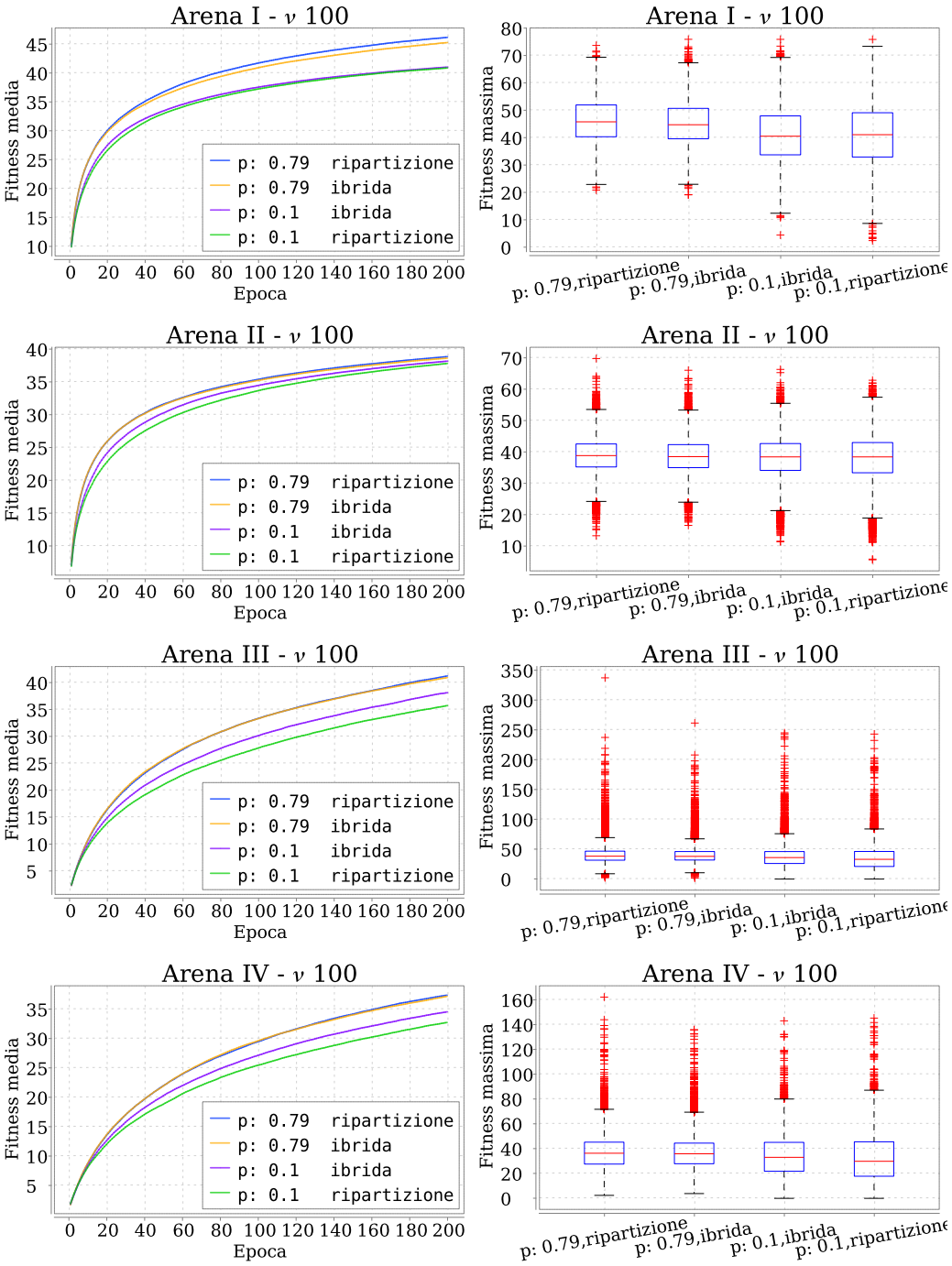


Figura 4.2: In figura sono mostrati gli andamenti medi di *fitness* e i valori massimi ottenuti per ogni robot. I dati sono stati raccolti utilizzando $\nu = 100$, i grafici sono suddivisi per arena.

4.2 Perturbazione iniziale

Per questo esperimento si utilizzano i risultati delle simulazioni discusse nel capitolo 3. Per ogni configurazione di parametri si prende l'un percento della popolazione che ha massimizzato il punteggio di *fitness* (popolazione d'*elite*), e per ogni individuo si estrae la rete booleana della sua miglior epoca. Per ogni rete ricavata in questo modo vengono eseguite 20 simulazioni, della durata di due epoche, con due soli robot dotati della rete booleana in questione. In 10 delle 20 simulazioni lo stato della rete booleana viene ripristinato a quello ricavato dalla miglior epoca del robot. Invece, nell'altra metà lo stato viene inizializzato casualmente per ogni robot della simulazione. Questo test serve per determinare se lo stato (insieme delle variabili booleane degli N nodi) sia parte integrante della rete booleana nel suo complesso. Se una volta avviata la simulazione la rete ricade nello stesso attrattore in cui ha ottenuto il punteggio di *fitness* massimo, si presuppone che il robot riesca a raggiungere livelli di *fitness* paragonabili a quello d'*elite*. Se invece la rete diverge da questo attrattore a causa del diverso stato iniziale, il robot potrebbe ottenere prestazioni molto inferiori. I robot a cui non è stato resettato lo stato iniziale dovrebbero sempre produrre un buon punteggio, o comunque, dovrebbero ottenere un risultato migliore rispetto ai robot a cui è stato resettato lo stato della rete booleana.

Si è deciso di eseguire questi test con due soli robot per volta, per evitare un'eccessiva competizione. In una situazione dove ogni robot è costituito da una rete booleana differente, e dove la fase di adattamento è ancora in corso, molto probabilmente la maggior parte dei robot è ferma o mostra un comportamento non ottimale, dunque, non vi è mai una grande competizione tra i robot nell'eseguire i task. In questo caso, invece, ogni robot è dotato della stessa rete booleana d'*elite*, provocando una forte competizione in quanto essi adottano le stesse strategie di movimento. Per ogni simulazione si considera il valore di *fitness* massimo che il robot ha ottenuto in una delle due epoche. I robot non applicano tecniche di adattamento tra le epoche, esse servono solo per dare all'agente la possibilità di ritornare in un regime di esecuzione ottimale durante la prima epoca, e di massimizzare il punteggio nella seconda.

I risultati ottenuti sono presentati nella figura 4.3 sotto forma di box-plot e suddivisi per arena. Per ogni configurazione (es. $\{p = 0.1,$ *ibrida*$\}$) è presente una coppia di box-plot: quello di sinistra indica il punteggio di *fitness* delle reti booleane a cui non è stato resettato lo stato iniziale, quello di destra i punteggi dei risultati ottenuti dai robot con stato iniziale casuale.

In tutte le configurazioni le coppie di box-plot sembrano raggiungere gli stessi livelli di *fitness*. Questo significa che indipendentemente dallo stato iniziale della rete booleana i robot riescono a tornare in un regime di funzionamento ottimale con la stessa probabilità. Con questo risultato è possibile affermare che i robot sono robusti rispetto ad una perturbazione totale dello stato della rete. Infatti, le reti booleane in questione riescono a tornare in un

regime di lavoro ottimale per l'esecuzione del task.

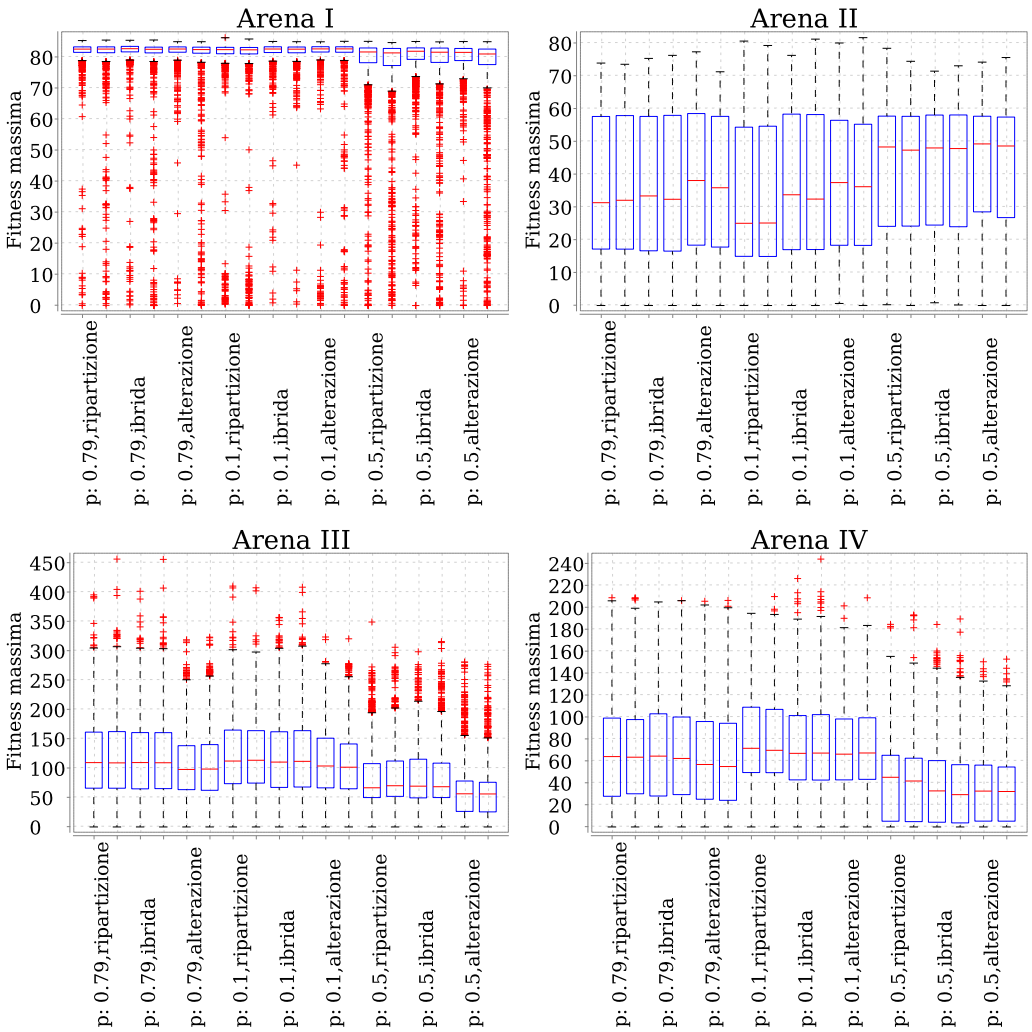


Figura 4.3: Per ogni configurazione sono disegnati due box-plot adiacenti: quello si sinistra riporta i valori massimi di *fitness* ottenuti dai robot con lo stato iniziale ripristinato, quello di destra i punteggi relativi ai robot con lo stato della rete booleana inizializzato casualmente.

Considerando che le reti booleane sono state estrapolate dai robot d'*elite* appartenenti agli esperimenti condotti nel capitolo 3, si auspicava che in queste simulazioni la maggior parte dei robot ottenessero un buon punteggio. Invece, come è possibile notare molti robot ottengono livelli di *fitness* vicini allo zero. Per mostrare meglio questo fenomeno si sono utilizzati dei grafici a box-plot simili a quelli precedenti, ma utilizzando i valori dei robot originari. Essi sono mostrati in figura 4.4. In questo caso il box-plot di sinistra riporta i livelli di *fitness* dei robot d'*elite* originari, mentre, il box-plot di destra riporta i livelli di *fitness* raggiunti dai robot appartenenti a quella metà a cui non è stato resettato lo stato iniziale. Nella prima arena i livelli di *fitness* sono paragonabili

a quelli dei robot originari. Mentre, è evidente che nelle altre tre arene molti robot non riescono a produrre un punteggio di *fitness* paragonabile a quello d'*elite*. Questo potrebbe essere causato dal diverso posizionamento iniziale del robot nell'arena. L'agente, non ritrovandosi nella stessa posizione rispetto al caso ottimo, non riesce a tornare in un regime di funzionamento ottimale. Dunque, i robot non possono essere considerati robusti rispetto ad un riposizionamento nell'arena. Un'altra causa potrebbe essere che il robot originario nella sua epoca migliore ha avuta una particolare fortuna nel posizionamento iniziale che gli ha permesso di raggiungere un livello di *fitness* tale da essere incluso nell'insieme dei robot d'*elite*. Tuttavia, questa conclusione non influenza il fatto che le reti booleane siano robuste ad una re-inizializzazione dello stato iniziale.

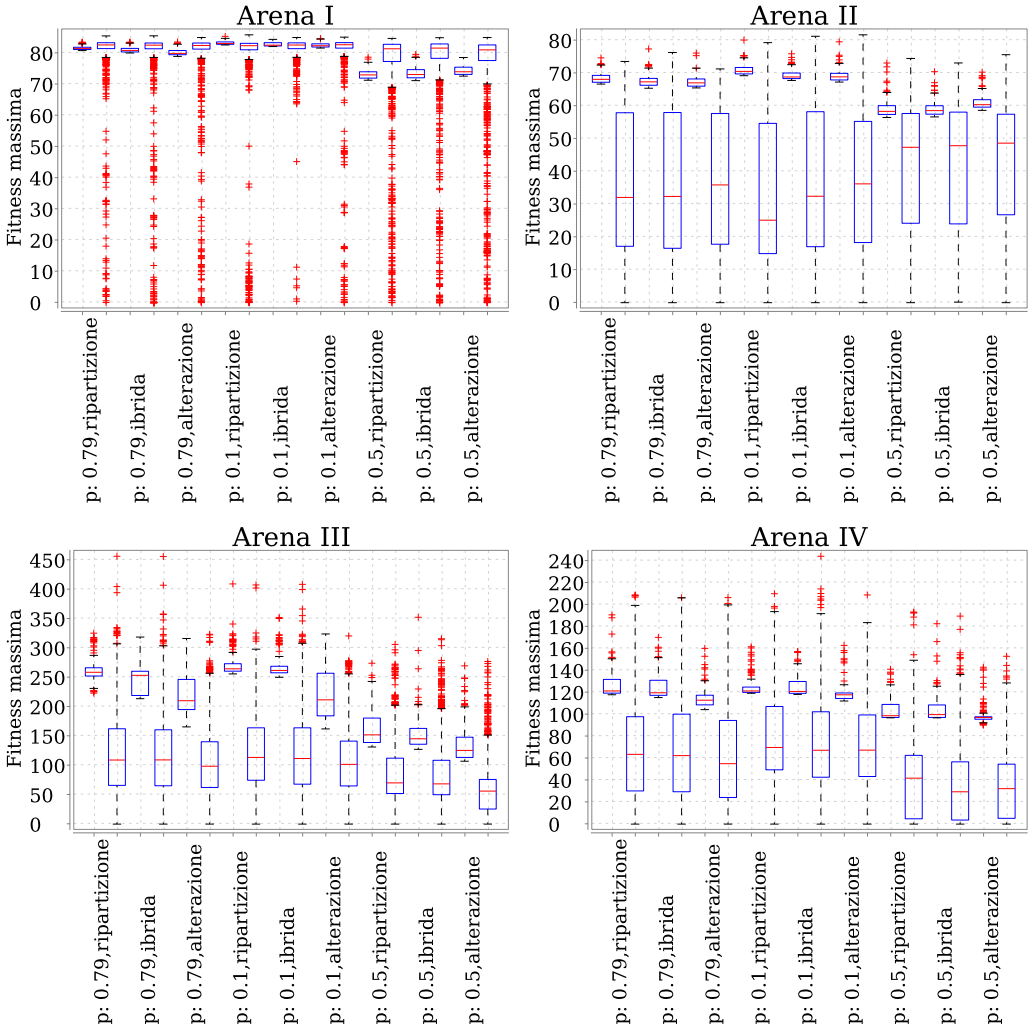


Figura 4.4: Per ogni configurazione sono disegnati due box-plot adiacenti: quello si sinistra riporta i valori di *fitness* dei 100 robot d'*elite* appartenenti alla configurazione sottostante, quello di destra i punteggi relativi ai robot con lo stato della rete booleana ripristinato al valore originario.

In conclusione, non è possibile affermare che il robot nella sua interezza sia robusto ad un riposizionamento casuale nell'arena. Quindi, il sistema robotico dipende dalla sua storia passata e non è sempre possibile cambiare il suo contesto di lavoro in modo trasparente. Invece, la sola rete booleana del robot riesce sempre a tornare in un regime di funzionamento ottimale, trattandosi di un sistema ergodico che non dipende dalla sua storia passata. Infatti, tutte le reti booleane testate sono robuste rispetto ad un diverso stato iniziale.

4.3 Malfunzionamenti

Questi esperimenti sono analoghi a quelli nella sezione precedente (4.2). In questo caso anziché re-inizializzare lo stato della rete, si mantiene quello originale, eliminando casualmente il 10% degli archi di ogni rete booleana sottoposta a malfunzionamenti. Questo test serve per determinare la robustezza delle reti nell'eventualità di guasti fisici. I risultati sono presentati come nel caso precedente con l'ausilio dei box-plot a coppie. Essi sono mostrati in figura 4.5. Anche in questo caso sono state effettuate 20 simulazioni per ogni rete booleana d'*elite* ricavata dagli esperimenti del capitolo 3. Nella metà di questi esperimenti vengono eliminati il 10% degli archi, nell'altra invece, le reti booleane rimangono immutate.

Con i risultati raccolti è possibile affermare che a seguito di malfunzionamenti, le reti critiche e caotiche anche nel caso più semplice della prima arena, subiscono un notevole deterioramento delle prestazioni. Le reti ordinate, invece, sembrano essere particolarmente resistenti a questo tipo di malfunzionamento. Le reti generate con $p = 0.1$ mostrano una particolare capacità nel mantenere prestazioni elevate anche subendo l'eliminazione del 10% degli archi. Questo fenomeno è particolarmente visibile nelle reti ordinate non sottoposte a mutazione (tecnica di adattamento *ripartizione*). Una possibile spiegazione potrebbe essere che le reti generate con $p = 0.79$ o $p = 0.5$ utilizzino una percentuale maggiore di nodi e archi per ottenere le dinamiche desiderate, e dunque, un malfunzionamento ha effetti più significativi rispetto a quelle ordinate. Le informazioni in queste tipologie di reti vengono memorizzate, propagate ed elaborate dall'intera rete booleana, e un cambiamento nella struttura causa una destabilizzazione di tutto il complesso. Le reti ordinate, invece, potrebbero essere caratterizzate da dinamiche interne strutturate in modo molto più ridondante, e una modifica della struttura interna non determina un cambiamento nel comportamento manifestato.

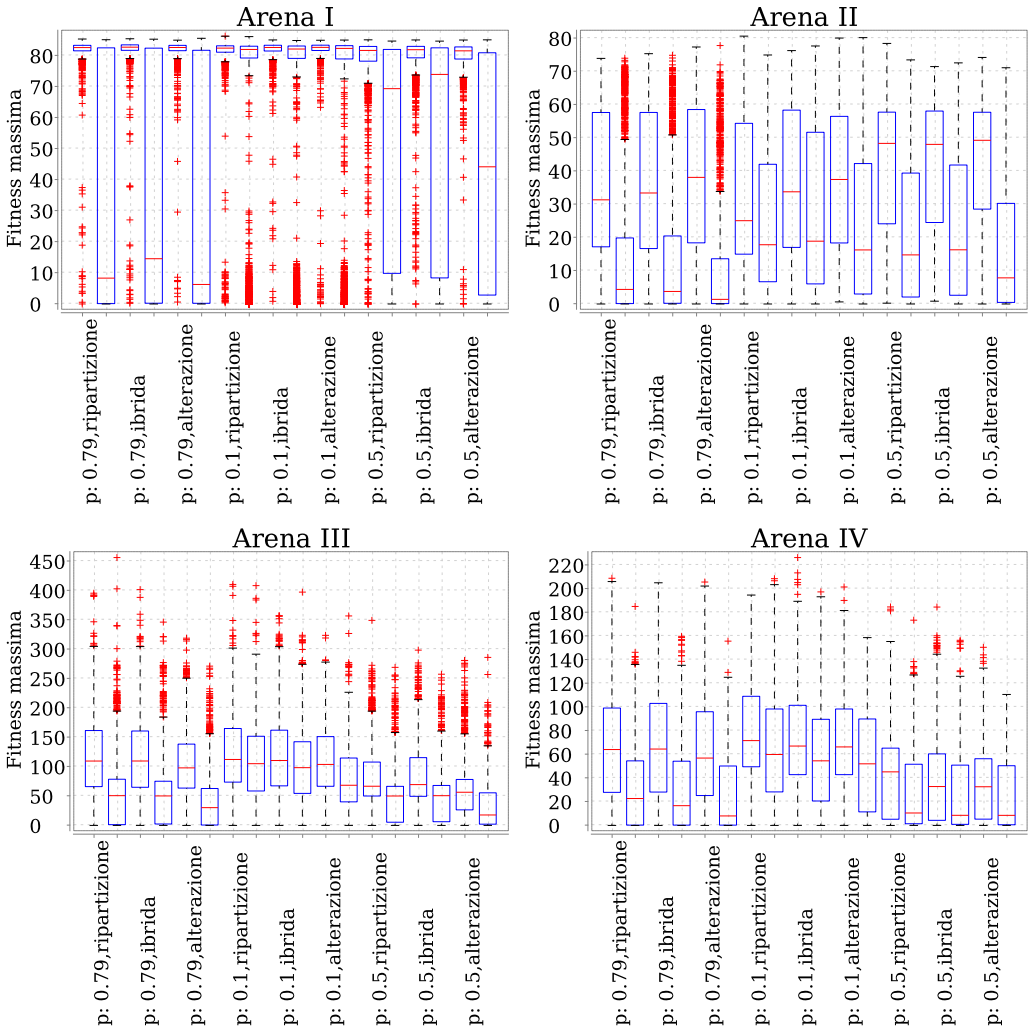


Figura 4.5: Per ogni configurazione sono disegnati due box-plot adiacenti: quello si sinistra riporta i valori massimi di *fitness* ottenuti dai robot con la rete booleana originaria, quello di destra i punteggi relativi ai robot con le reti booleane danneggiate.

4.4 Conclusioni

Grazie all'analisi condotta in questo capitolo è possibile asserire che le reti booleane utilizzate sono robuste a perturbazioni dello stato. In particolare, si sono testate due forme di perturbazioni: una continua durante la fase di adattamento, e una totale in cui si resetta lo stato della rete booleana all'inizio della simulazione. In entrambi i casi si sono ottenuti ottimi risultati, privilegiando leggermente l'uso delle reti booleane generate con $p = 0.79$. Questi risultati confermano l'ipotesi iniziale, infatti, le reti booleane in un regime critico sono caratterizzate da un'adattabilità e una robustezza superiore rispetto a quelle in regimi ordinati o caotici.

Per quanto riguarda l'alterazione forzata della struttura della rete (eliminazione di archi), si sono dimostrate particolarmente robuste le reti di natura ordinata, indipendentemente dall'arena e dalla tecnica di adattamento utilizzata. Questo risultato, seppur contrario all'ipotesi iniziale, non dimostra che le reti critiche e caotiche non possano essere robuste rispetto a questo tipo di malfunzionamento. Infatti, l'eliminazione degli archi avviene dopo la fase di adattamento. In uno studio futuro si potrebbe provare a sottoporre le reti booleane a malfunzionamenti durante la fase di adattamento. Questo esperimento servirebbe per determinare se a fronte di tale evenienza le reti booleane riescano comunque a adattarsi producendo un buon punteggio.

Nel prossimo capitolo si studia il comportamento dei robot usando una misura tipica della teoria dell'informazione. Inoltre, si tenta di guidare l'adattamento dei robot attraverso funzioni obiettivo indipendenti dal task, e basate sul livello di entropia generato dal movimento dei robot.

Capitolo 5

Entropia come fitness

In questo capitolo si esplora la possibilità di usare funzioni obiettivo basate sull'entropia. Inizialmente, si descrive una forma di misurazione dell'entropia nel campo della teoria dell'informazione: la *Shannon Entropy*. Successivamente, viene fatta un'analisi sui dati raccolti dalle simulazioni del capitolo 3. Si calcola l'entropia sui valori sensoriali dei robot e si mette in relazione con i punteggi di *fitness* ottenuti. Infine, si propone una forma di adattamento che sfrutta l'analisi effettuata, definendo una funzione obiettivo ausiliaria, generica e indipendente dal task. Questa analisi è motivata da una una questione importante nell'evoluzione naturale e artificiale. Essa concerne la relazione tra complessità e *fitness* [10]. Infatti, la definizione di funzioni obiettivo indipendenti dal task può essere paragonata all'introduzione di istinti intrinseci per lo sviluppo di comportamenti utili. Questi istinti, durante la fase di adattamento, permettono di esplorare lo spazio di ricerca più liberamente, senza essere vincolati da una descrizione esplicita della soluzione desiderata [14].

5.1 Shannon Entropy

L'entropia di Shannon [13] è una misura chiave nell'ambito della teoria dell'informazione. Essa quantifica il grado d'incertezza proveniente da una sorgente d'informazione. La definizione è la seguente:

$$H(X) = -\sum_{x \in X} P(x) log_2(P(x)) \tag{5.1}$$

dove X è la sorgente d'informazione con dominio finito e discreto, x è un possibile simbolo della sorgente. $P(x)$ è la probabilità che dalla sorgente X scaturisca il simbolo x. Intuitivamente, all'aumentare di $H(X)$ cresce il grado d'incertezza della sorgente, viceversa, se $H(X)$ tende a 0 significa che la sorgente X emette simboli facilmente prevedibili. Una sorgente che emette un unico simbolo ha entropia 0, una sorgente che emette n simboli con probabilità uniforme ha entropia massima $log_2(n)$.

5.2 Analisi dell'entropia

In questa particolare analisi si è misurata l'entropia proveniente dalle letture dei sensori di prossimità dei robot. In particolare, la sorgente d'informazione X è composta dai valori utilizzati per perturbare la rete booleana. Si ricorda che il robot, dotato di 24 sensori di prossimità, elabora otto valori booleani ad ogni passo di simulazione. Il vocabolario della sorgente X è quindi composto da parole di otto bit, con un totale di 256 possibili simboli. La figura 5.1 mostra un esempio di stato in cui un robot potrebbe trovarsi in un determinato istante della simulazione.

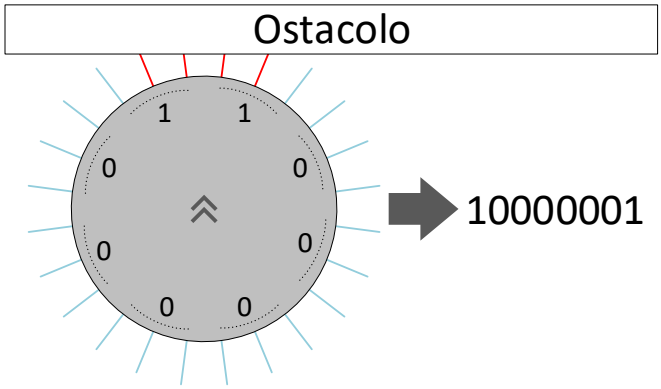


Figura 5.1: Esempio di un robot che incontra un ostacolo. La sequenza binaria generata dai sensori di prossimità nel passo di simulazione dell'esempio è 10000001. Quest'ultima viene utilizzata per perturbare la rete booleana sugli otto nodi di input. Queste sequenze sono i simboli del vocabolario che vengono usate per calcolare i valori di entropia generati dai vari robot.

Per ogni simulazione viene estrapolata l'epoca migliore di ciascun robot (epoca in cui ha raggiunto il maggior valore di *fitness*). Successivamente, viene estratta la sequenza di letture dei sensori di prossimità per tutti gli 800 passi di un'epoca, ottenendo così una sequenza di 800 simboli per ciascun robot. Per calcolare $H(X)$ è sufficiente determinare la probabilità dei simboli presenti nell'insieme contando il loro numero di occorrenze.

Sono stati generati alcuni grafici a dispersione che mettono in relazione i punteggi di *fitness* dei robot e la loro entropia. In questi grafici, suddivisi per tipologia di task, si mostra la quantità d'incertezza che i robot hanno incontrato nel percorso durante la loro miglior epoca. Nei grafici 5.2, 5.3, 5.4, 5.5 sono raffigurati i risultati ottenuti.

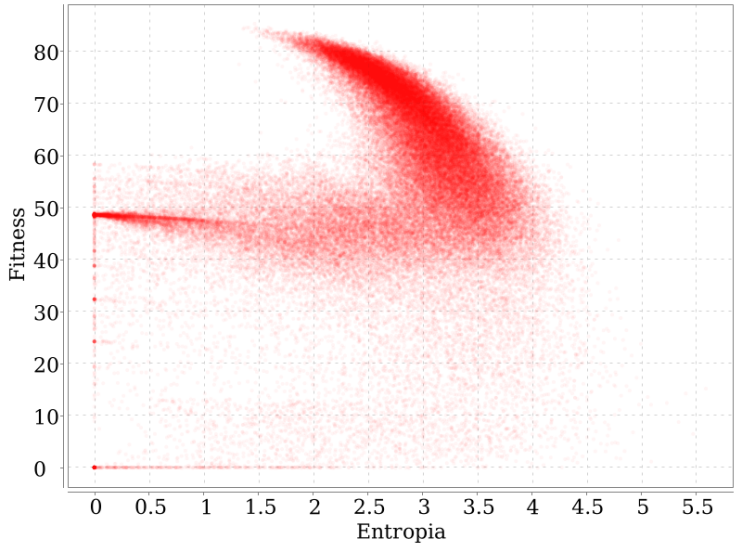


Figura 5.2: Grafico a dispersione che mette in relazione i punteggi di *fitness* con il livello di entropia calcolato sulle letture dei sentori di prossimità. L'asse delle ordinate rappresenta il punteggio di *fitness* raggiunto dai robot nelle loro migliori epoche. L'asse delle ascisse rappresenta il valore di entropia ricavato dalla *Shennon Entropy* sulla base delle letture di prossimità effettuate dai robot nella loro miglior epoca. I risultati sono riferiti alle simulazioni svolte nella prima arena. Ogni punto corrisponde ad un singolo robot.

Nel caso del grafico 5.2 sembra esserci una forte relazione tra entropia e *fitness*. È utile ricordare che questo esperimento è riferito all'arena rettangolare, dove l'unico task dei robot è quello di muoversi in linea retta evitando gli ostacoli. Guardando il grafico è possibile notare che vi è un *cluster* di valori posizionato in entropia 0 e *fitness* 50. Questo fenomeno è riconducibile ad un particolare comportamento che permette di raggiungere valori di *fitness* elevati sfruttando una caratteristica della funzione obiettivo. I robot dotati di questo comportamento si muovono in linea retta per un passo di simulazione, e ruotano a sinistra o a destra per il passo seguente. Il risultato è quello di ottenere robot che ruotano in cerchio, evitando di esplorare l'arena, e allo stesso tempo accumulano valori di *fitness* non trascurabili (circa il 50% del massimo). La conseguenza è che i robot in questione ottengono un valore pressoché nullo di entropia, in quanto, ruotando su sé stessi, non incontrano particolari novità nella loro ristretta regione di movimento. Questo risultato ha permesso di trovare una tipologia di comportamenti che sfrutta una debolezza della funzione obiettivo. Infatti, dal punto di vista del progettista, l'intento è quello di far muovere il robot in linea retta, e farlo ruotare solo in caso di collisione imminente. Tuttavia, alcuni robot si sono adattati muovendosi in una regione di spazio molto limitata. Grazie a questa analisi è stato possibile individuare la presenza di questi comportamenti non desiderati.

Un'altra analisi del grafico sembra indicare che i robot migliori siano quelli

che si sono imbattuti in un grado d'incertezza medio: da 1.4 a 2.5. È possibile individuare una regione a cono dove entropia e *fitness* convergono, e l'inclinazione del cono suggerisce che al diminuire dell'entropia il punteggio migliori. Questa caratteristica può essere spiegata tramite questo ragionamento: una volta che il robot riesce a dotarsi di una buona strategia per muoversi, esso raggiungerà il punteggio massimo quando riesce ad applicarla senza incontrare imprevisti. È da tenere in considerazione che durante la simulazione sono presenti dieci robot, e questi potrebbero intralciarsi a vicenda durante il compimento del task. Nei casi fortuiti in cui il robot è caratterizzato da una buona strategia di movimento, e non trova nessun ostacolo imprevisto, esso riesce a raggiungere un punteggio elevato, riscontrando un'entropia inferiore rispetto ai casi in cui si imbatte in ostacoli imprevisti. Dunque, i robot che si sono adattati con più successo, sono quelli che hanno raggiunto valori elevati di *fitness*, minimizzando anche l'entropia sugli input.

Le considerazioni fatte valgono anche per il grafico 5.3.

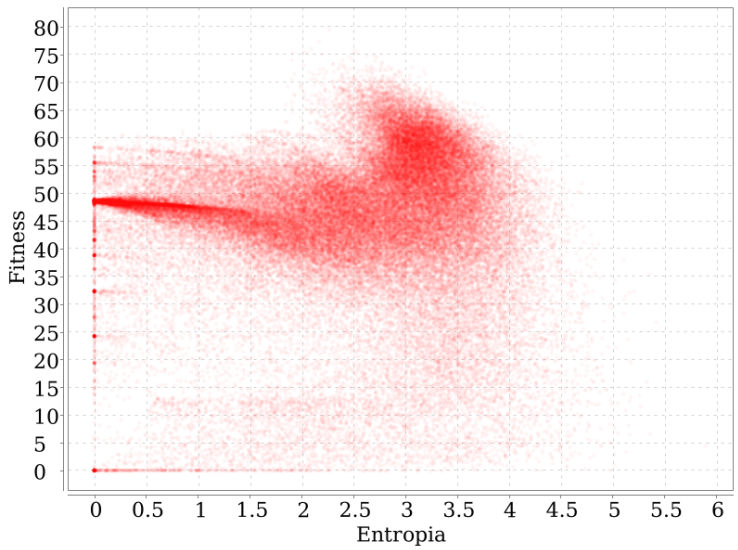


Figura 5.3: Grafico a dispersione che mette in relazione i punteggi di *fitness* al livello di entropia calcolato sulle letture dei sentori di prossimità. I risultati sono riferiti alle simulazioni svolte nella seconda arena.

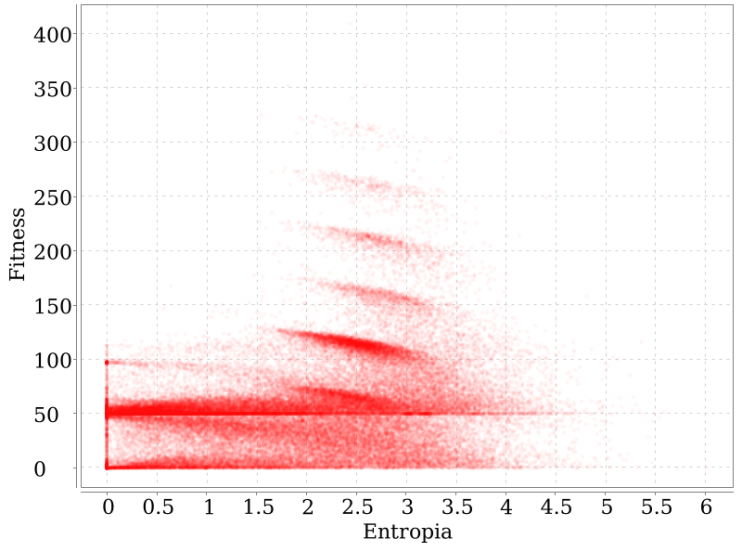


Figura 5.4: Grafico a dispersione che mette in relazione i punteggi di *fitness* al livello di entropia calcolato sulle letture dei sentori di prossimità. I risultati sono riferiti alle simulazioni svolte nella terza arena.

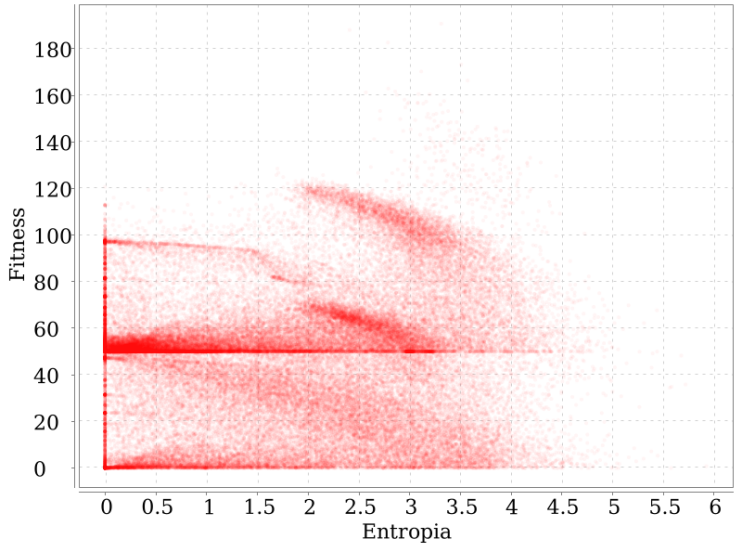


Figura 5.5: Grafico a dispersione che mette in relazione i punteggi di *fitness* al livello di entropia calcolato sulle letture dei sentori di prossimità. I risultati sono riferiti alle simulazioni svolte nella quarta arena.

Anche nei casi dei grafici 5.4 e 5.5, il cui risultato deriva da task sostanzialmente più complessi (*foraging*), è evidente che i robot con punteggio più elevato sono caratterizzati da un'entropia media rispetto ai valori limite. In particolare, nel caso del grafico 5.4, riferito al task di *foraging* nella terza arena, è possibile notare circa 6 livelli orizzontali. Queste separazioni nette sono

causate dal sistema a premi della funzione obiettivo del task di *foraging*. Infatti, per ogni raccolta e deposito il robot ottiene 50 punti. Anche in questo caso ogni livello, essendo inclinato, suggerisce una relazione inversamente proporzionale tra entropia e *fitness* come nel primo caso. Questo significa che a parità di consegne effettuate, sono migliori, in termini di *fitness*, quei robot che sono riusciti a diminuire la sorpresa sugli input, e dunque, a muoversi in maniera più consapevole, imbattendosi in meno ostacoli, e accumulando un punteggio superiore.

5.3 Applicazione

Disponendo di questi risultati si tenta di definire una funzione obiettivo indipendente dal task, e guidare l'adattamento dei robot sfruttando anche le informazioni ricavate dall'analisi precedente. L'idea è quella di porre al robot un'entropia obiettivo, e di usare la distanza da essa come funzione obiettivo da minimizzare. Il robot, al termine di ogni epoca, determina se mantenere o meno le modifiche applicate al software di controllo sulla base della distanza dal valore obiettivo, e non più sulla base del punteggio di *fitness* accumulato. Le tecniche di adattamento viste in precedenza rimangono invariate, cambia solo il modo in cui si valutano i robot. La distanza dall'entropia obiettivo viene poi messa in relazione con il punteggio di *fitness* calcolato con le formule dei task originari, tuttavia, i punteggi non vengono mai usati dai robot durante la fase di adattamento. La funzione da minimizzare diventa:

$$(H' - H(X))^2 \tag{5.2}$$

dove H' è l'entropia obiettivo predefinita, $H(X)$ è l'entropia calcolata alla fine di ogni epoca, X è la sorgente di simboli relativi ai sensori di prossimità.

In questi esperimenti sono state testate tre differenti entropie obiettivo: $0.5, 2.2, 6.0$. Sono state usate reti booleane generate con $p = 0.1$ e $p = 0.79$, utilizzando tutte e tre le tecniche di adattamento discusse in precedenza (*ripartizione*, *alterazione*, *ibrida*). I restanti paramenti delle simulazioni sono descritti nell'appendice A. Sono state effettuate un totale di mille simulazioni per ogni configurazione testata. In seguito all'analisi effettuata in precedenza, il risultato atteso è che i robot guidati dall'entropia obiettivo $H' = 2.2$ riescano a raggiungere valori più elevati di *fitness*. In figura 5.6 sono mostrati i risultati ottenuti utilizzando questa nuova funzione obiettivo basata sull'entropia. Tutti gli esperimenti sono stati svolti nella prima arena, usando come riferimento la funzione obiettivo del primo task. Ogni punto del grafico è riferito ai risultati di un singolo robot ed è generato per l'epoca in cui esso ha minimizzato la distanza dalla propria entropia obiettivo.

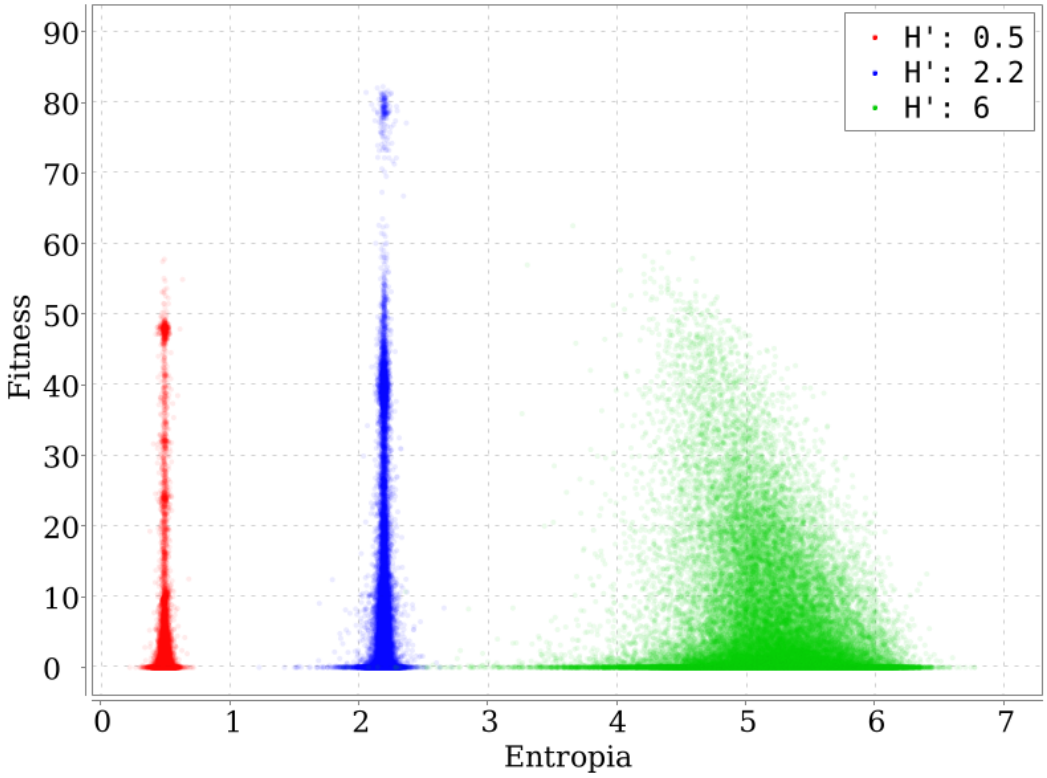


Figura 5.6: Risultati dell'adattamento basato su un valore di entropia obiettivo. Il grafico a dispersione mette in relazione l'entropia dei sensori di prossimità e il punteggio di *fitness* che avrebbe ottenuto il robot. Per ogni agente viene generato un punto che rappresenta l'epoca in cui ha ottenuto la distanza minima dall'entropia obiettivo. La fitness viene calcolata al termine della simulazione con la formula del primo task.

È evidente che i risultati migliori, in termini di *fitness*, derivano da quei robot con entropia obiettivo 2.2. Questo è esattamente il risultato che ci si aspetta, infatti, i robot migliori nell'analisi precedente ottengono valori di entropia da 1.4 a 2.5. Ciò significa che è possibile determinare delle funzioni obiettivo indipendenti e propedeutiche al task da compiere. Inoltre, è utile distinguere la differenza tra entropia obiettivo 0.5 e 6.0. Analizzando visivamente il comportamento dei robot si è scoperto che usare un obiettivo di entropia elevato è preferibile ad un obiettivo vicino allo zero. I robot con obiettivo 0.5 manifestano comportamenti prevedibili, ad esempio, girare su s'è stessi o stare completamenti fermi. Quelli con un obiettivo di entropia elevato, invece, manifestano comportamenti più complessi dal punto di vista di un osservatore esterno. Infatti, si muovono esplorando l'intera arena cercando di massimizzare l'insieme di stimoli distinti percepiti. Essendo che l'obiettivo finale è quello di far muovere il robot in linea retta evitando gli ostacoli, è evidente che fissare $H' = 6.0$ produce robot con comportamenti molto più desiderabili rispetto ad $H' = 0.5$.

5.3.1 Funzione obiettivo ibrida

Il passo seguente è quello di combinare le due tecniche usando la funzione obiettivo originale in concomitanza ad una funzione basata sull'entropia. Per realizzare questo esperimento è necessario definire una funzione obiettivo complessiva che il robot utilizzerà per guidare l'adattamento. Come nel caso precedente l'esperimento sarà svolto nella prima arena, utilizzando la funzione obiettivo descritta nell'equazione 2.1. Questa funzione deve tenere conto anche di un'entropia obiettivo a cui il robot deve sottostare. A tal fine si è deciso di moltiplicare il punteggio di *fitness* risultante dall'equazione 2.1 per il valore ricavato dalla funzione $h(X)$. Quest'ultimo viene calcolato al termine di ogni epoca sulla base di $H(X)$ e di H'. La funzione $h(X)$ è definita nel seguente modo:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.3}$$

$$\sigma'(x) = \frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \tag{5.4}$$

$$h(X) = \begin{cases} 4 \cdot \sigma'(\alpha \cdot (H(X) - H')), & \text{se } H(X) < H' \\ 4 \cdot \sigma'(\beta \cdot (H(X) - H')), & \text{se } H(X) \geq H' \end{cases} \tag{5.5}$$

dove:

- $\sigma(x)$ è la funzione Sigmoid e $\sigma'(x)$ la sua derivata.
- H' è l'entropia obiettivo.
- $H(X)$ è l'entropia calcolata al termine dell'epoca sulla base delle letture effettuate sui sensori di prossimità.
- α e β sono due parametri che permettono di tarare la funzione h.

Impostando $H' = 1.7$, $\alpha = 3$ e $\beta = 0.75$ si ottiene una curva che premia i robot che ottengono un'entropia nell'intorno di 1.7, penalizzando più velocemente i robot che si discostano da questo valore verso il limite inferiore. La curva è mostrata in figura 5.7.

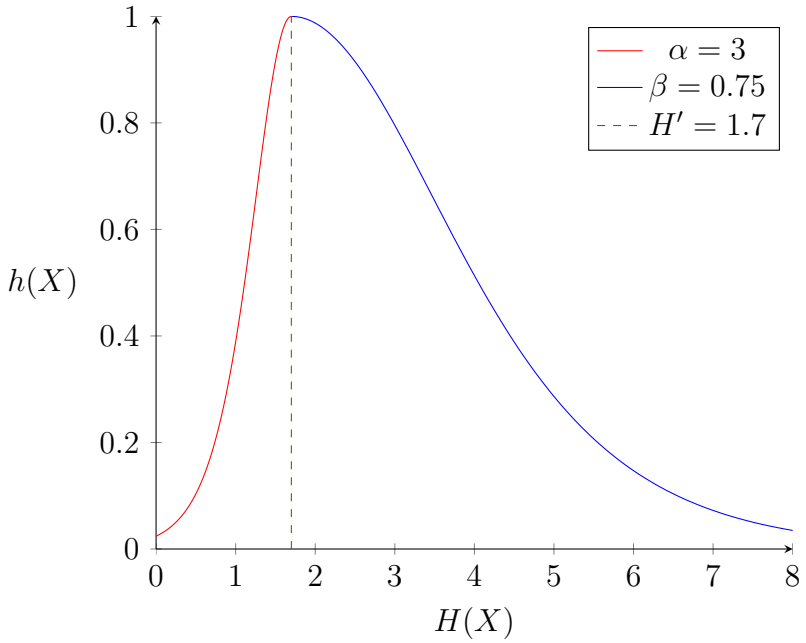


Figura 5.7: Funzione $h(X)$. Il valore ricavato viene usato come moltiplicatore per calcolare il punteggio di *fitness* nella funzione obiettivo ibrida.

Il punteggio di *fitness* finale è quindi calcolato moltiplicando il valore ottenuto da $h(X)$ per il punteggio ricavato dall'equazione 2.1. In questo modo i robot per adattarsi devono sicuramente ottenere un valore di entropia maggiore di 0, risolvendo l'uso dell'*exploit* della funzione di fitness discusso in precedenza. Allo stesso tempo devono sottostare alla funzione obiettivo originale, cercando di imbattersi in un grado d'incertezza medio.

Come nel caso precedente sono state usate reti booleane generate con $p = 0.1$ e $p = 0.79$, utilizzando tutte e tre le tecniche di adattamento (*ripartizione*, *alterazione*, *ibrida*). I restanti paramenti delle simulazioni sono descritti nell'appendice A. Tutti gli esperimenti sono svolti nella prima arena con un totale di mille ripetizioni per configurazione. In figura 5.8 è mostrato il grafico a dispersione dei valori di entropia e *fitness* raggiunti dai robot. Vi è una notevole somiglianza con i risultati iniziali, tuttavia, sembra essere scomparso il *cluster* di punti vicino al valore di entropia zero. Questo significa che i robot non manifestano più il comportamento indesiderato in cui sfruttavano la funzione obiettivo per ottenere un punteggio elevato pur non esplorando l'arena. I risultati raccolti non mostrano un particolare vantaggio rispetto al solo utilizzo della formula 2.1, se non per la risoluzione dell'*exploit* descritto in precedenza. Inoltre, il punteggio di *fitness* medio calcolato su tutte le migliori epoche è leggermente inferiore. Una possibile spiegazione è che la funzione $h(X)$ non consente ai robot di esplorare in modo efficiente lo spazio di configurazioni con le tecniche di adattamento utilizzate. Infatti, come si può notare dalla figura 5.2 sembrerebbe che i robot prima di raggiungere livelli di *fitness* elevati, debbano transitare per valori di entropia superiori a 2, per poi iniziare

la risalita. In questa situazione, invece, i robot sono penalizzati sin da subito se ottengono un valore di entropia non conforme all'andamento di $h(X)$. In un lavoro futuro si potrebbe provare a stabilire un moltiplicatore in funzione dell'entropia e del punteggio di *fitness* ottenuto. Ad esempio, $h(X, F)$ dove F è il punteggio di *fitness* al termine dell'epoca.

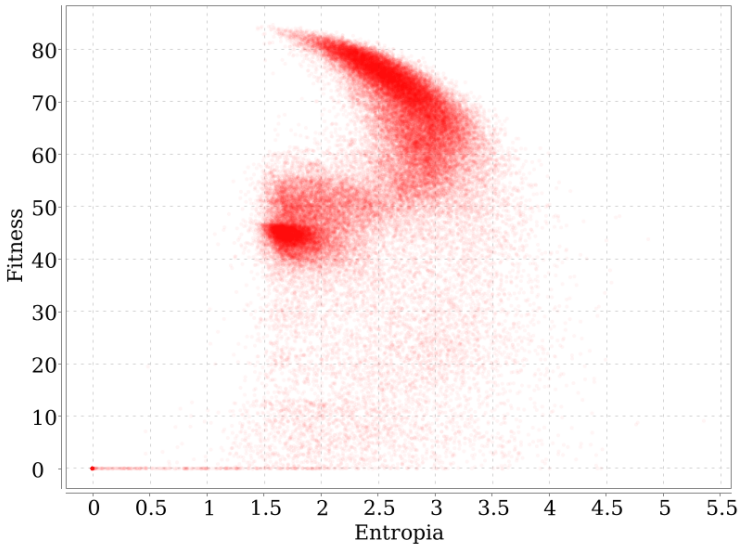


Figura 5.8: Risultati dell'adattamento basato sulla funzione obiettivo ibrida: il punteggio viene calcolato moltiplicando il valore di *fitness* ricavato dalla formula 2.1 per il moltiplicatore $h(X)$. Ogni punto del grafico è generato per l'epoca in cui un robot ha massimizzato il valore della funzione obiettivo ibrida. L'asse delle ordinate rappresenta il punteggio di *fitness* puro e non moltiplicato per $h(X)$.

Conclusioni

Questo progetto ha permesso di rispondere a tutti i quesiti formulati inizialmente. Lo studio si focalizza sul determinare quale tipologia di rete booleana è più efficacie come software di controllo di un sistema robotico. Inoltre, ci si interroga su quale tecnica di adattamento sia più consona, ovvero, se sia necessario modificare la struttura interna delle reti booleane, o se è sufficiente modificarne l'interfacciamento con il robot.

L'ipotesi principale è che le reti booleane critiche siano dotate di tutte le caratteristiche computazionali necessarie per massimizzare la prestazione del robot. Dunque, questa tipologia di rete booleana non necessità di una riconfigurazione interna, ma è sufficiente trovare un buon interfacciamento con i sensori e attuatori del robot. Per esplorare questa ipotesi sono stati definiti vari ambienti virtuali in cui i robot schierati devono massimizzare una certa funzione obiettivo. Il risultato prodotto da questa funzione è detto "punteggio di *fitness*" e viene usato per confrontare le prestazioni dei robot.

Dai dati raccolti si possono constatare evidenti differenze tra i risultati prodotti dalle tre tipologie di reti booleane (ordinate, critiche, caotiche). I robot dotati di una rete booleana caotica ottengono risultati mediamente inferiori, indipendentemente dalla tecnica di adattamento utilizzata. Mentre, i risultati migliori derivano dalle reti ordinate e critiche. Si è constatato che in task semplici le reti ordinate ottengono un particolare vantaggio rispetto a quelle critiche. Tuttavia, vi è una tendenza a preferire reti booleane critiche all'aumentare della difficoltà del task da compiere. Inoltre, utilizzando reti critiche, è preferibile applicare la tecnica di adattamento *ripartizione* che non introduce modifiche alla struttura della rete, ma soltanto una riconfigurazione dei nodi di input e output. Questo primo risultato è conforme all'ipotesi iniziale.

Successivamente, si sono studiati gli effetti della tecnica di *alterazione* che introduce modifiche alla struttura interna della rete. In questo caso lo scopo è quello di determinare gli effetti delle modifiche sulle dinamiche interne delle reti. Una rete booleana casuale non può essere classificata in ordinata, critica o caotica dopo esser stata sottoposto ad una fase di evoluzione. Tuttavia, si possono studiare a posteriori le caratteristiche paragonandole a quelle delle reti booleane casuali non sottoposte a mutazione. In particolare, si è usato il valore di Derrida che misura il livello di propagazione di una perturbazione, e il numero di stati visitati dalla rete. Si è scoperto che l'uso della tecnica di *alterazione*, su robot dotati di reti booleane ordinate, produce dinamiche

interne simili a quelle di reti booleane critiche. Ci si concentra sulle reti ordinate perché sono quelle che hanno prodotto ottimi risultati insieme a quelle critiche. Questo risultato è un ulteriore supporto all'ipotesi iniziale. Infatti, le reti booleane critiche producono ottimi risultati con la tecnica *ripartizione*, mentre, le reti booleane ordinate producono ottimi risultati con l'applicazione di entrambe le tecniche contemporaneamente, producendo reti booleane con caratteristiche simili a quelle critiche.

Un'ulteriore analisi è rivolta alla robustezza delle reti booleane utilizzate. In particolare, si è misurata la robustezza a due tipi di perturbazioni: una continua durante la fase di adattamento, e una iniziale su reti booleane già evolute. In entrambi i casi, sia le reti ordinate che quelle critiche, sono risultate particolarmente robuste rispetto a queste tipologie di perturbazioni. Nel caso della perturbazione continua si è dimostrato come questi sistemi riescano ad assorbire un notevole rumore esterno, adattandosi comunque all'ambiente ed al task. Nel caso della perturbazione iniziale, si è dimostrato come questa forma di controllo riesca con buon successo a tornare in un regime di funzionamento ottimale. Infatti, una nuova inizializzazione dello stato non comporta una perdita di prestazioni. Successivamente, si è monitorato l'andamento delle prestazioni dopo aver rimosso casualmente alcuni archi dalle reti booleane già evolute. Il risultato ottenuto è che le reti ordinate sono le uniche ad essere robuste rispetto a questo tipo di malfunzionamento.

Infine, si sono messe in relazione le prestazioni ottenute dai robot con una misura tipica della teoria dell'informazione. Si è scoperto che i robot migliori, in termini di *fitness*, ottengono un valore di entropia medio rispetto ai valori limite. Dopodiché, si è sperimentata una funzione obiettivo basata proprio sulle misure di entropia effettuate dai robot. I risultati raccolti indicano che può essere particolarmente utile definire una funzione obiettivo sulla base di una misura indipendente dal task da compiere. Questo permette di conferire ai robot istinti utili allo sviluppo di comportamenti propedeutici ai task da compiere.

Appendice A

Parametri delle simulazioni

La configurazione di una simulazione viene definita attraverso i seguenti parametri.
Parametri generali:

argos arena da utilizzare. Varia in base all'esperimento specifico (I, II, III o IV).

ticks_per_seconds passi di simulazione al secondo ($\frac{1}{\Delta t}$). Fissato a 10.

experiment_length durata dell'intero esperimento, espresso in secondi. Fissato a 16000s.

epoch_length durata di un'epoca, espressa in secondi. Fissato a 80s in modo da ottenere 200 epoche per esperimento.

robot_count robot schierati nell'arena. Fissato a 10.

max_wheel_speed massima velocità delle ruote dei robot in cm/s. Fissato a 25.

wheels_nodes nodi di output dedicati all'attivazione dei motori. Fissato a 2.

proximity_threshold soglia di binarizzazione dei segnali provenienti dai sensori di prossimità. Fissato a 0.1.

proximity_nodes nodi di input dedicati ai sensori di prossimità. Fissato a 8.

light_threshold soglia di binarizzazione dei segnali provenienti dai sensori di luminosità. Fissato a 0.1.

light_nodes nodi di input dedicati ai sensori di luminosità. Varia in base all'arena in cui viene schierato il robot (0 nella I e II, 8 nella III e IV).

Parametri di generazione delle RBN:

N nodi della rete. Fissato a 100.

K grado entrante di ogni nodo. Fissato a 3.

p bias di generazione delle funzioni booleane. Varia in base all'esperimento specifico $(0.1, 0.5, 0.79)$.

self_loops abilita la presenza di *self loop* nella rete. Fissato a *False*.

only_distinct_connections indica se ogni nodo deve avere K nodi entranti distinti. Fissato a *True*.

override_outputs_p effetto descritto nella sezione 2.3.2. Fissato a 0.5.

allow_io_node_overlap indica se è possibile che un nodo di input sia usato anche come nodo di output e viceversa. Fissato a *False*.

Parametri di alterazione:

connection_rewires numero di archi mutati per epoca. Varia in base all'esperimento specifico (3 se l'alterazione è attiva, 0 altrimenti).

self_loops indica se la mutazione di archi può produrre *self loop* nella rete. Fissato a *False*.

only_distinct_connections indica se la mutazione di archi deve mantenere il vincolo in cui ogni nodo ha K nodi entranti distinti. Fissato a *True*.

function_bit_flips numero di bit da alterare nelle tavole di verità delle funzioni booleane ad ogni epoca. Varia in base all'esperimento specifico (8 se l'alterazione è attiva, 0 altrimenti).

Parametri di ripartizione:

input_rewires numero di nodi di input da rimpiazzare. Varia in base all'esperimento specifico (2 se la ripartizione è attiva, 0 altrimenti).

output_rewires numero di nodi di output da rimpiazzare. Varia in base all'esperimento specifico (1 se la ripartizione è attiva, 0 altrimenti).

allow_io_node_overlap indica se è possibile che dopo la fase di ripartizione un nodo di input sia usato anche come nodo di output e viceversa. Fissato a *False*.

Un sentito ringraziamento alla mia famiglia per avermi sostenuto durante tutto il percorso universitario.

Bibliografia

[1] Maximino Aldana, Susan Coppersmith, and Leo P Kadanoff. Boolean dynamics with random couplings. *Perspectives and Problems in Nolinear Science*, pages 23–89, 2003.

[2] Michael Bonani, Valentin Longchamp, Stéphane Magnenat, Philippe Rétornaz, Daniel Burnier, Gilles Roulet, Florian Vaussard, Hannes Bleuler, and Francesco Mondada. The marxbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4187–4193. IEEE, 2010.

[3] Michele Braccini. *Towards a Boolean network-based Computational Model for Cell Differentiation and its applications to Robotics*. PhD thesis, alma, Aprile 2020.

[4] B Derrida and G Weisbuch. Evolution of overlaps between configurations in random boolean networks. *Journal de physique*, 47(8):1297–1303, 1986.

[5] Bernard Derrida and Yves Pomeau. Random networks of automata: a simple annealed approximation. *EPL (Europhysics Letters)*, 1(2):45, 1986.

[6] Christoph Fretter, Agnes Szejka, and Barbara Drossel. Perturbation propagation in random and evolved boolean networks. *New Journal of Physics*, 11(3):033005, 2009.

[7] Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.

[8] Massimo Pigliucci et al. *Phenotypic plasticity: beyond nature and nurture*. JHU Press, 2001.

[9] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari. On the design of Boolean network robots. In C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, J. Merelo, F. Neri, M. Preuss, H. Richter, J. Togelius, and G. Yannakakis, editors, *Applications of Evolutionary Computation*, volume 6624, pages 43–52. 2011.

[10] Andrea Roli, Antoine Ligot, and Mauro Birattari. Complexity measures: open questions and novel opportunities in the automatic design and analysis of robot swarms. *Frontiers in Robotics and AI*, 6:130, 2019.

[11] Andrea Roli, Marco Villani, Alessandro Filisetti, and Roberto Serra. Dynamical criticality: overview and open questions. *Journal of Systems Science and Complexity*, 31(3):647–663, 2018.

[12] Volkan Sevim and Per Arne Rikvold. Chaotic gene regulatory networks can be robust against mutations and noise. *Journal of theoretical biology*, 253(2):323–332, 2008.

[13] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[14] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Mutual information as a task-independent utility function for evolutionary robotics. In *Guided Self-Organization: Inception*, pages 389–414. Springer, 2014.

[15] Agnes Szejka and Barbara Drossel. Evolution of canalizing boolean networks. *The European Physical Journal B*, 56(4):373–380, 2007.