ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

ANALISI DELL'APPLICAZIONE DI EMPOWERMENT A ROBOT SOTTOPOSTI AD ADATTAMENTO IN AMBIENTE DINAMICO

Elaborato in
SISTEMI INTELLIGENTI ROBOTICI

Relatore
Prof. ANDREA ROLI

Co-relatore
Dott. MICHELE BRACCINI

Presentata da
MANUEL BONARRIGO

Terza Sessione di Laurea
Anno Accademico 2020 – 2021

PAROLE CHIAVE

Adattamento evolutivo online

Plasticità fenotipica

Reti booleane random

Empowerment

Argos

Indice

Elenco delle figure

Introduzione

Nello sviluppo di sistemi intelligenti robotici, l'approccio più comune per trasferire all'*automaton* un modello della conoscenza che gli permetta di valutare in maniera autonoma l'efficacia e la bontà delle proprie azione risiede nell'utilizzo di funzioni obiettivo specifiche al task che si desidera esso svolga. Per quanto efficaci, al cuore dell'utilizzo di tali funzioni risiede il problema della poca riusabilità delle stesse nel momento in cui si debba affrontare un cambiamento, che sia nell'obiettivo specifico o nell'ambiente in cui il robot viene calato: in questi casi è necessario rivalutare e riformulare l'intero problema, scrivendo delle nuove funzioni obiettivo.

La soluzione proposta da un gruppo di ricerca dell'università di Hertfordshire consiste nello sviluppo di una funzione obiettivo *universale*, basata sulla qualsiasi forma di accoppiamento sensori–attuatori all'interno del robot, denominata *empowerment*.

Il lavoro svolto analizza l'efficacia dell'applicazione dell'empowerment come funzione obiettivo innestata sull'adattamento intrapreso da robot la cui struttura di controllo sia costituita da reti booleane di tipo criticale. Tale efficacia viene misurata sia sostituendo la qualsiasi funzione obiettivo *task-specific* con l'empowerment in degli esperimenti ad arena singola, sia applicando una commistione delle due funzioni in esperimenti ad arena multipla, in cui l'ambiente si considera cambiare in seguito a modalità e cause non note al robot.

Nel primo capitolo si introducono i concetti di rete booleana ed empowerment, rispettivamente la base della forma di controllo dei robot simulati negli esperimenti, e la metrica analitica i cui effetti sull'adattamento online sono stati analizzati.

Nel secondo capitolo si espone il modello del dominio studiato e la struttura del framework realizzato per affiancare il ricercatore e permettere uno svolgimento agile, controllato e strutturato delle simulazioni.

Nel terzo capitolo si mostrano gli esperimenti condotti e relative varianti, esplicitando i principali risultati quantitativi ottenuti.

Nel quarto capitolo vengono presentati i risultati qualitativi degli esperimenti effettuati, e si delineano una serie di possibili sviluppi futuri.

In coda all'elaborato sono presenti due appendici, mirate ad espandere rispettivamente dei concetti aggiuntivi del design del framework e mostrare alcuni risultati grafici dei migliori percorsi effettuati dai robot durante le simulazioni.

Capitolo 1

Stato dell'arte

In questo capitolo si introducono i concetti teorici sui quali si sono basate le sperimentazioni in analisi. In sezione 1.1 si introducono la teoria e il meccanismo dietro il funzionamento delle reti booleane, e del concetto di *criticalità*; in sezione 1.2 viene introdotto il concetto di *empowerment* e il suo funzionamento.

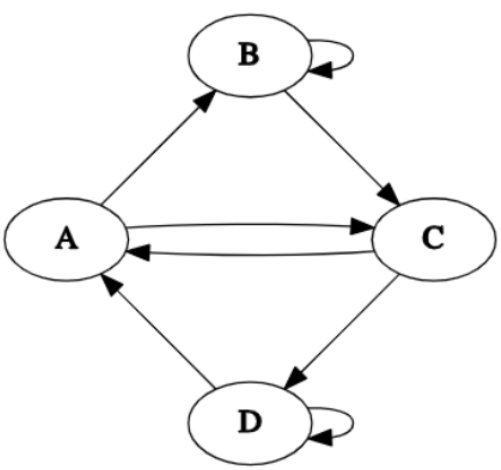
1.1 Reti booleane

Le reti booleane sono state introdotte da Stuart Kauffman nel 1969 come modello matematico per le reti di regolazione genica. [6] In questa sezione verranno introdotte le caratteristiche strutturali e funzionali di una rete booleana, e successivamente in che modo vengono applicate in un contesto di controllo robotico.

1.1.1 Descrizione del modello

Le reti booleane rappresentano attraverso un grafo orientato parzialmente connesso il modello di un sistema dinamico, discreto nel tempo e nello stato

in cui gli elementi all'interno dei nodi possono assumere i valori binari 0 o 1, e attraverso un altrettanto numero di funzioni booleane, che determinano l'evoluzione dei valori al progredire del passo temporale, basandosi sul valore dei nodi collegati come entranti. Le informazioni necessarie a descrivere completamente la topologia di una rete booleana sono quindi il numero n di nodi che compongono la rete booleana ed il numero k di connessioni in entrata di ogni nodo.



C	D	A
0	0	0
0	1	1
1	0	1
1	1	1

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

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

C	D	D
0	0	0
0	1	1
1	0	0
1	1	0

Figura 1.1: Esempio minimale di rete booleana, con $n = 4$ e $k = 2$. [9]

La *traiettoria* di una rete booleana è definita come la sequenza di stati interni che essa assume durante il proprio ciclo di vita. Quando sia il primo elemento della traiettoria, che la configurazione delle tabelle della verità che contengono le funzioni booleane, che ancora le connessioni che intercorrono fra i nodi sono scelte a caso, si parla di RBN, *random boolean network*: in questo caso si introduce un concetto denominato *bias*, o b, un terzo parametro di costruzione che rappresenta la probabilità con la quale assegnare il valore 1

in fase di costruzione delle funzioni booleane.

Formalizzando il modello, in una rete booleana costituita da n nodi, a ognuno dei quali si associano i valori booleani $x_i, i = 1, ..., n$, sono possibili 2^n configurazioni interne. Le tabelle di verità, che determinano il valore di un nodo al momento $t+1$ sulla base dei k valori entranti al momento t, assumono quindi la forma di funzioni $f_i(x_{i_1}, ..., x_{i_k})$, il cui insieme di valori possibili ha una cardinalità pari a 2^{2^k}. Il singolo istante temporale di una traiettoria, ovvero lo stato della rete al tempo t si può definire come la lista dei valori dei nodi al tempo t, formalmente $s(t) = (x_1(t), ..., x_n(t))$, il chè definisce formalmente una traiettoria come $T = (s(t_1), ..., s(t_{max}))$.

1.1.2 Ordine, caos e criticalità

È stato dimostrato come, avendo fissato il parametro n, è possibile sfuttare una formula nota [4] per conoscere un determinato valore del parametro k a partire dal valore imposto di b in maniera tale da ottenere tre regimi di funzionamento diversi delle RBN, dove con regime si intende la capacità di reagire all'influenza degli stimoli esterni.

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

In un regime *ordinato* la rete tende ad avere una scarsa propagazione degli stimoli ricevuti in input, facendo fatica a deviare la propria traiettoria dallo stato precedente.

In un regime *caotico* la rete tende ad essere dominata da una forte instabilità, che impedisce di reagire in maniera robusta alle perturbazioni esterne.

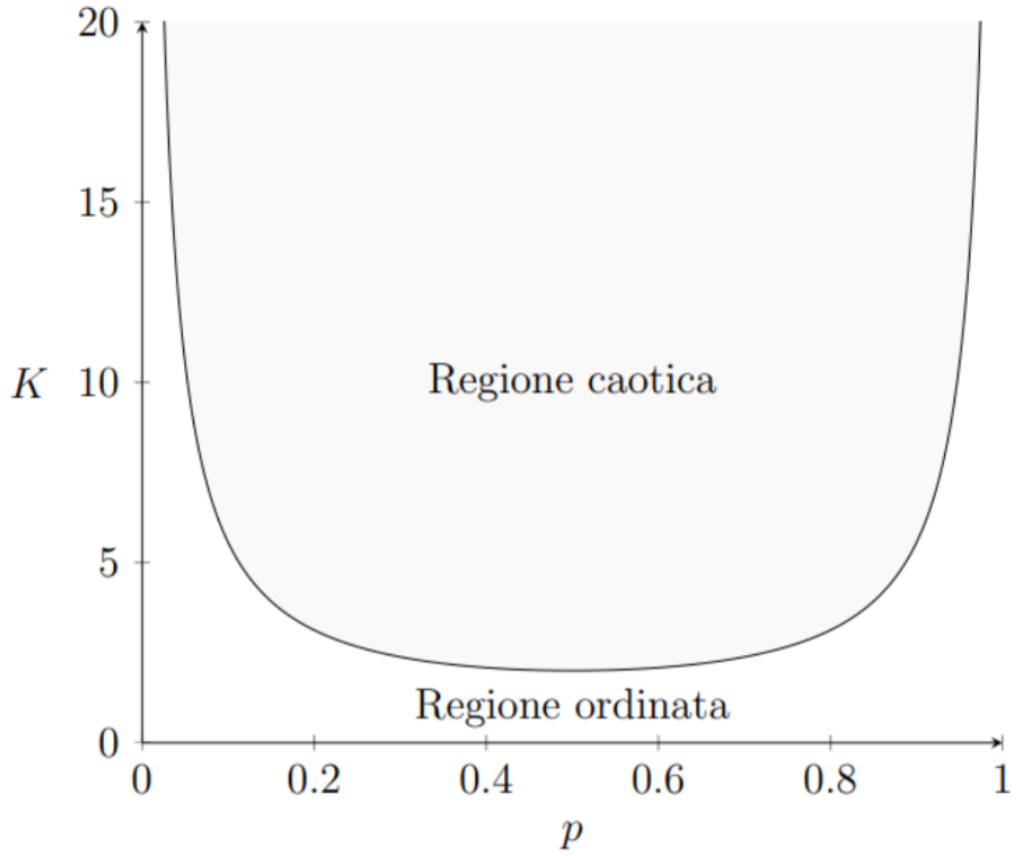


Figura 1.2: Confine tra regime ordinato e caotico [10]

Il confine fra le due aree in figura 1.2 è definito regime *criticale*, che in base ad una congettura nota come *criticality hypothesis* è la chiave per ottenere reti che siano in grado di esprimere un equilibrio tra robustezza ed adattabilità [11].

Sfruttando il confine definito nella formula 1.1 è possibile quindi determinare i valori critici con la quale inizializzare una rete booleana. Nel caso di questa trattazione, in cui k è rimasto fissato a 3, i valori critici di b risultano essere 0.21 e 0.79.

1.1.3 RBN come controller robotico

Le reti booleane possono essere utilizzate come struttura di controllo robotico [4]. Previa opportuna codifica in valore booleano delle percezioni registrate dal robot tramite i propri sensori, è possibile immettere tali valori in una serie prestabilita di nodi di input: il cambiamento dall'esterno dei valori di tali nodi porterà ad una perturbazione dei valori interni della rete, modificando il valore dei nodi stabiliti essere di output. Decodificando il valore dei nodi di output è possibile quindi attivare i corrispettivi attuatori. Nella figura 1.3 è possibile visualizzarne una rappresentazione grafica.

In quello che viene definito come adattamento *online*, la topologia della rete si considera inizializzata con una struttura che non cambierà per l'intera esistenza dell'*automaton*. In questa maniera l'unico aspetto variabile è il modo in cui le funzioni booleane permettono o meno la propagazione di perturbazioni esterne fino ai nodi di output.

Mantenere la stessa topologia pur avendo la capacità di esprimere comportamenti diversi è una proprietà nota con il nome di *plasticità fenotipica* [4]. Il concetto, preso in prestito dalla genetica, denota la capacità di esprimere fenotipi differenti sulla base dello stesso patrimonio genetico in funzione dell'ambiente in cui l'individuo è calato: nel parallelo, il robot (individuo) controllato da rete booleana (genotipo) modula il proprio apparato sensoriale (fenotipo) in base agli stimoli percepiti dall'ambiente.

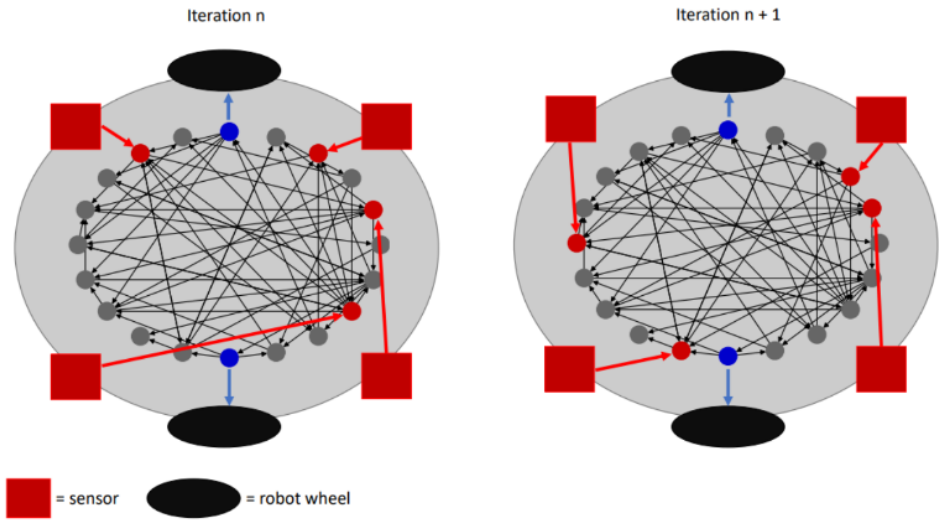


Figura 1.3: Meccanismo di controllo tramite rete booleana

1.2 Empowerment

Nello sviluppo di sistemi intelligenti robotici, l'approccio più comune per trasferire all'*automaton* un modello della conoscenza che gli permetta di valutare in maniera autonoma l'efficacia e la bontà delle proprie azione risiede nell'utilizzo di funzioni obiettivo specifiche al task che si desidera esso svolga. Per quanto efficaci, al cuore dell'utilizzo di tali funzioni risiede il problema della poca riusabilità delle stesse nel momento in cui si debba affrontare un cambiamento, che sia nell'obiettivo specifico o nell'ambiente in cui il robot viene calato: in questi casi è necessario rivalutare e riformulare l'intero problema, scrivendo delle nuove funzioni obiettivo.

La soluzione proposta da un gruppo di ricerca dell'università di Hertfordshire [7] consiste nello sviluppo di una funzione obiettivo *locale* ed *universale*, basata sulla qualsiasi forma di accoppiamento sensori–attuatori all'interno del robot, denominata *empowerment*. La località della funzione consiste nel non

necessitare di una conoscenza globale del mondo, per estensione sia fisica che temporale, in quanto fornisce un feedback tarato sull'individuo; l'universalità deriva dal potersi applicare a qualsiasi individuo, di qualsiasi specie, calato in qualsiasi ambiente.

L'idea alla base è quella di creare un modello di calcolo che associ la percezione del qualsiasi stimolo esterno alla massimizzazione delle azioni che è possibile intraprendere di conseguenza: questa è infatti la lettura comportamentale che Klyubin et al. danno come motivazione fondamentale intrinseca alla base di qualsiasi *branch* decisionale. Riportando l'esempio della pubblicazione in cui viene presentato l'empowerment, un batterio preferirà trovarsi in punti dove riesce a percepira una concentrazione di sostanze nutrienti più alta, poichè questa condizione gli permetterà maggiori possibilità in termini di sopravvivenza e riproduzione. [7]

L'empowerment si pone quindi come obiettivo di misurare la capacità di controllo che un individuo possiede nei confronti dell'ambiente in cui è immerso, esprimendo tale capacità unicamente in relazione con ciò che viene percepito dall'individuo stesso, scollegandosi dalla visione globale di un qualsiasi osservatore.

1.2.1 Definizione formale di empowerment

Tale definizione viene effettuata sfruttando una serie di concetti provenienti dalla teoria dell'informazione: la definizione di un canale di comunicazione, la misura di *mutual information* e la capacità di canale.

Canale di comunicazione Consideriamo un mittente e un ricevente, che comunicano su un canale di comunicazione discreto e privo di memoria. Identifichiamo il segnale inviato dal mittente tramite la variabile aleatoria X e il

segnale, possibilmente diverso da quello originale, percepito dal ricevente attraverso la variabile aleatoria Y. Il canale usato per comunicare determina la corrispondenza tra il segnale inviato e quello ricevuto; nel caso di segnali discreti, questo canale può essere descritto dalla distribuzione condizionale $p(y|x)$.

Mutual Information La *mutual information* misura la quantità di informazione contenuta nel segnale ricevuto proveniente dal messaggio inviato. La formula per calcolarla si definisce in funzione della distribuzione caratteristica del canale $p(y|x)$ e della distribuzione di probabilità del segnale trasmesso $p(x)$:

$$I(X;Y) = \sum_{X,Y} p(y|x)p(x) \log_2 \frac{p(y|x)}{\sum_X p(y|x)p(x)} \tag{1.2}$$

La *mutual information* è simmetrica e acausale, in quanto non richiede la conoscenza delle distribuzioni a priori dei due segnali.

Capacità di canale La capacità di canale viene definita come la massima *mutual information* del canale, calcolata su tutte le possibili distribuzioni del segnale trasmesso:

$$C(p(y|x)) = \max_{p(x)} I(X;Y) \tag{1.3}$$

Contrariamente alla *mutual information*, la capacità di canale è asimmetrica e causale, poichè richiede il controllo completo della variabile X.

La capacità di un canale discreto arbitrario può essere stimata in modo efficiente tramite l'algoritmo iterativo di Blahut-Arimoto [3][2].
Per descrivere l'empowerment, consideriamo un robot dotato di un certo numero di sensori S e attuatori A, calato in un ambiente dalla morfologia e perturbazioni possibili sconosciute definito come R. Definiamo inoltre come t un qualsiasi istante temporale del sistema, e di conseguenza S_t, A_t e R_t come lo stato di sensori, attuatori e l'ambiente al tempo t.

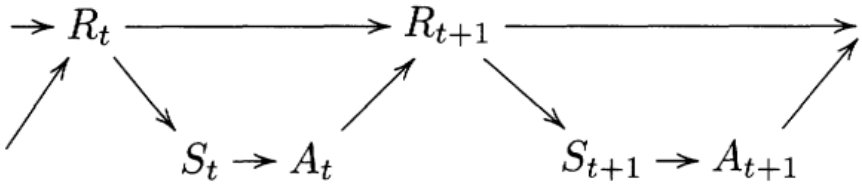


Figura 1.4: Modello dei rapporti causali fra sensori, attuatori e resto del mondo.

Sfruttando le variabili appena descritte è possibile costruire una rete Bayesiana che metta in mostra le dinamiche temporali e causali del sistema: al tempo t, le condizioni ambientali R_t determinano i valori dei sensori S_t i quali, a loro volta, determinano lo stato degli attuatori A_t i quali, a loro volta, avranno un impatto sul mondo esterno, conducendolo nello stato R_{t+1}. Guardando alla rete definita nei termini del problema della comunicazione su canale, si possono interpretare le azioni intraprese dagli attuatori del robot come causa principale di iniezione di modifiche nell'ambiente, e di conseguenza come sorgente delle percezioni sensoriali del robot stesso. In questa visione il robot possiede un senso di feedback riguardo al proprio controllo sul mondo. L'ambiente diventa quindi il canale di comunicazione filtrante di un mittente rappresentato dagli attuatori che inviano il proprio messaggio ai sensori del robot stesso.

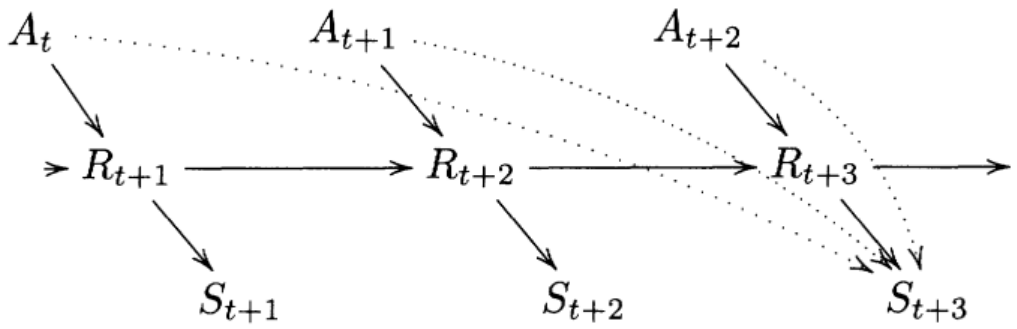


Figura 1.5: Modello dei rapporti causali fra sensori, attuatori e resto del mondo.

Introducendo come n la sequenza di azioni che il robot deliberatamente decidesse di intraprendere al tempo t, ci si chiede quanta informazione sarebbe stata trasmessa al tempo $t + n$. Rimodellando i rapporti di causa-effetto che intercorrono tra gli stati dei sensori, degli attuatori e dell'ambiente tramite una rete causale (figura 1.5), possiamo denotare la sequenza di n azioni decise al tempo t con la variabile aleatoria $A_t^n = (A_t, A_{t+1}, \ldots, A_{t+n})$ e i suoi possibili valori con a_t^n; descriviamo lo stato dei sensori dopo l'esecuzione delle n azioni con la variabile aleatoria S_{t+n} e i suoi possibili valori con s_{t+n}; infine, i rapporti causali tra le variabili in oggetto vengono descritti tramite la distribuzione condizionale $p(s_{t+n}|a_t)$, che rappresenta il canale di comunicazione.

Definiamo quindi l'*empowerment* come la capacità del canale di comunicazione che ha attuatori del robot come sorgente e i suoi sensori come destinazione:

$$\mathfrak{E}_t = C(p(s_{t+n}|a_t^n)) = \max_{p(a_t^n)} I(a_t^n; S_{t+n}) \tag{1.4}$$

Questa quantità è misurata in bit: si ottiene un valore uguale a 0 se il robot non riesce a percepire di aver esercitato alcun controllo sull'ambiente o, ana-

logamente, se il robot non riesce a trasmettere informazione dai suoi attuatori ai suoi sensori tramite l'ambiente descritto dal canale di comunicazione.

In generale, massimizzare l'*empowerment* significa:

- cercare di raggiungere quelle parti del mondo in cui le azioni dell'agente riescono a determinare al meglio gli stati sensoriali;
- modificare i sensori e gli attuatori dell'agente in modo da migliorare il controllo dell'empowerment.

Capitolo 2

Ambiente Sperimentale

In questo capitolo si analizzeranno le tecnologie impiegate e sviluppate durante lo sviluppo dell'elaborato di tesi. Nella sezione 2.1 si analizza la struttura e le capacità del simulatore utilizzato, in sezione 2.2 si presenta il framework realizzato per assistere ed automatizzare il più possibile ogni fase della sperimentazione.

2.1 ARGoS

ARGoS (*Autonomous Robots Go Swarming*)[1] è un simulatore multirobot open-source, le cui qualità di rilievo sono l'essere stato dimostrato particolarmente efficiente pur riuscendo a mantenere una flessibilità tale da potersi dichiarare general purpose. [8] La flessibilità è stata ottenuta introducendo una modularità del design interno tale da permettere all'utente di riscrivere intere parti del sistema per accomodare le esigenze del particolare esperimento nel caso in cui tali necessità non fossero già soddisfatte dal simulatore stesso.

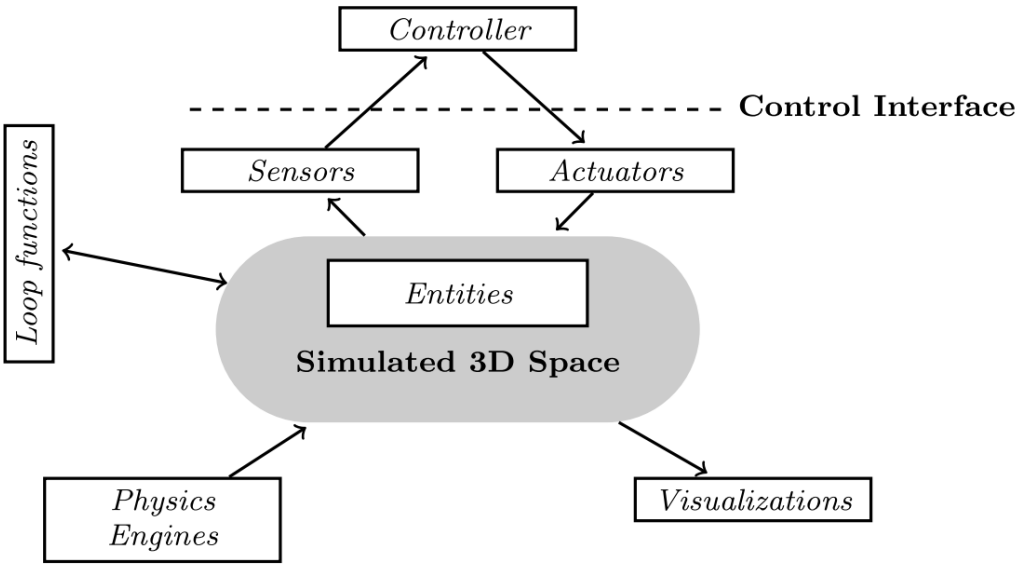


Figura 2.1: L'architettura di ARGoS. I riquadri in bianco corrispondono ai moduli ridefinibili dall'utente[8]

Perchè sia possibile utilizzare ARGoS è necessario fornire una descrizione dello spazio 3D che deve essere simulato, e perchè il robot faccia qualcosa oltre che ad esistere, anche una forma di controller: questi infatti sono stati gli unici due moduli (rimandando alla figura 2.1) con i quali si è entrati in contatto per effettuare le sperimentazioni.

Le informazioni necessarie a creare una rappresentazione interna dell'ambiente vengono fornite tramite un file XML, attraverso il quale si popola il simulatore con le entità di base necessarie a ricreare le impostazioni sperimentali e si associano i controller realizzati ai robot coinvolti. La struttura minimale di tale file è espressa nel listato 2.1. Leggendo il file di configurazione, ARGoS si preoccupa di "montare" sui robot i sensori e attuatori definiti ed assegnati, ed associare specifici controller a specifici robot nel caso si stiano trattando problemi con una moltitutidine di esemplari.

```
<argos-configuration>
  <framework>
    <experiment></experiment>
  </framework>
  <controllers>
    <lua_controller>
      <actuators></actuators>
      <sensors></sensors>
      <params/>
    </lua_controller>
  </controllers>
  <arena></arena>
  <physics_engines></physics_engines/>
  <visualization></visualization>
</argos-configuration>
```

Listato 2.1: Struttura di primo livello minimale di un file di configurazione *.argos*, privato degli attributi e della descrizione delle entità che popolerebbero l'ambiente.

2.1.1 Ambiente

Quello che nel listato 2.1 viene definito come elemento `<arena>` racchiude le informazioni necessarie a generare la nicchia ambientale all'interno della quale si intendono calati i robot sotto analisi. Oltre alla possibilità di specificarne le dimensioni, si possono determinare le proprietà degli elementi con i quali i robot si troveranno ad interagire. Per effettuare gli esperimenti in analisi al capitolo 3 si sono utilizzate entità già presenti nel dominio ambientale del

simulatore, opportunamente combinate per generare la nicchia rilevante.

- **Ostacolo -** Il concetto di ostacolo solido è costituito da un parallelepipedo del quale si possono impostare la dimensioni, e decidere se essere semovibile. Tramite questo componente si è strutturato il perimetro di tutte le arene sperimentali, e nell'esperimento *Navigazione fra ostacoli* si è utilizzato per costituire l'arena in analisi.

- **Luce -** Una sfera luminosa al cui centro si considera posizionata la sorgente del segnale, di cui è possibile specificare colore ed intensità. Viene utilizzata nell'esperimento *Ricerca spaziale assistita* per guidare l'adattamento del robot.

- **Terreno -** È possibile specificare un'immagine che il simulatore considererà come conformazione del terreno. Il concetto di terreno è stato utilizzato nell'esperimento di *Ricerca spaziale assistita* e nell'esperimento di *Navigazione su tracciato*.

2.1.2 Componenti

Per ognuna delle componenti ambientali descritte in sezione 2.1.1 sono stati utilizzati degli analoghi sensori in grado di rilevarne presenza e modificazioni. Questi sono resi disponibili al controller tramite l'interfaccia `Argos.robot` e strutturati come in figura 2.2. Nello specifico sono stati adoperati sensori di:

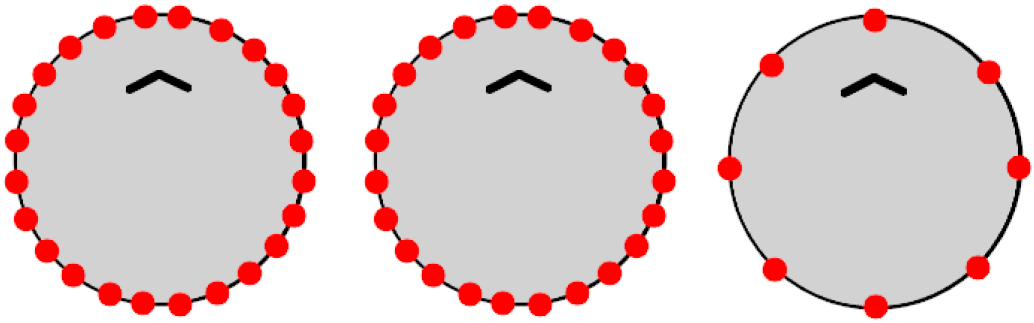
- **Prossimità -** Disposti intorno al robot, rilevano la presenza di ostacoli entro un raggio di 10 centimetri, ritornando un valore nell'intervallo $[0, 1]$ in base alla vicinanza con l'ostacolo rilevata.

- **Luminosità -** Disposti intorno al robot, ritornano un valore nell'intervallo $[0, 1]$ sulla base dell'intensità della luce rilevata.

- **Terreno -** Disposti intorno al robot, individuano zone bianche o nere sul terreno, rispondendo con valore 1 nel caso in cui sia rilevata una zona di terreno bianca, 0 nel caso in cui sia rilevata una zona nera.

- **Posizione -** Un sensore logico che permette di conoscere in maniera assoluta le coordinate del robot all'interno dell'arena.

Inoltre, come unica forma di attuazione, sono state utilizzate le ruote messe a disposizione del *footbot* (figura 2.3), attivabili separatamente ed a differente velocità.



(a) Disposizione dei sensori di luminosità (b) Disposizione dei sensori di prossimità (c) Disposizione dei sensori di terreno

Figura 2.2: Disposizione dei sensori "fisici" messi a disposizione dall'interfaccia *robot* di ARGoS

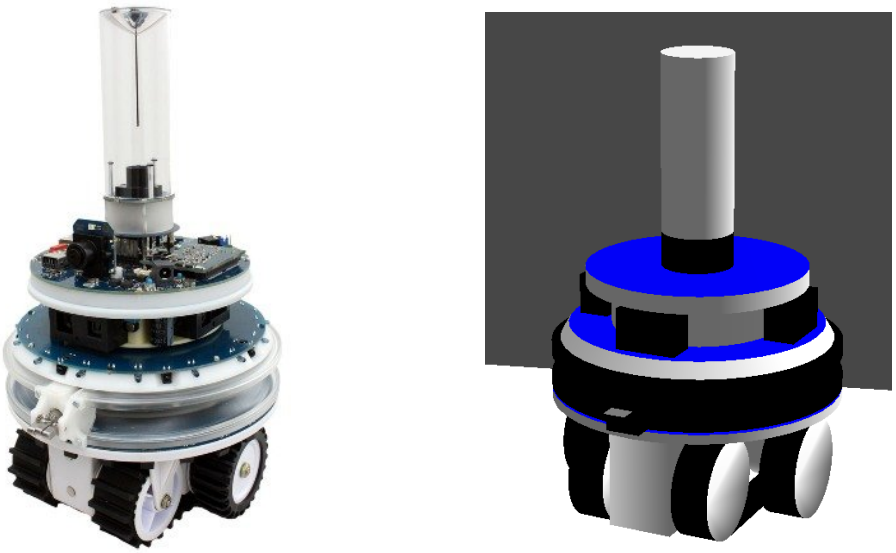
2.1.3 Controller

Realizzare un controller adeguato ad essere integrato nel funzionamento di ARGoS consiste nell'implementare la semplice interfaccia definita nel listato 2.2. Le fasi di *init* e *destroy* implementano rispettivamente le necessità di inizializzazione dell'ambiente di controllo e di finalizzazione della fase simulata; la fase di *step* viene richiamata ad ogni passo della fase di simulazione di

ARGoS, ed è necessaria ad aggiornare le variabili di controllo e ad attuare le strategie di movimento stabilite.

```
function init()    end
function step()    end
function destroy() end
```

Listato 2.2: Struttura dell'interfaccia *controller* che ARGoS richiede rispettata in relazione ad ogni robot coinvolto



(a) Un footbot reale (b) Un footbot simulato tramite ARGoS

Figura 2.3: La concretizzazione fisica e virtuale del Footbot

2.2 Framework sviluppato

Il framework realizzato in maniera parallela al lavoro sperimentale ha avuto come obiettivo principale la sedimentazione in codice dei passi necessari a

completare un'iterazione del processo sperimentale. L'espansione della *codebase* è quindi avvenuta di pari passo con il consolidamento della conoscenza dell'ambiente sperimentale e delle necessità funzionali ed analitiche del processo, progredendo da una mera *utility* di esecuzione ad un software integrato che ha inglobato praticamente tutte le responsabilità automatizzabili del lavoro svolto.

Lo sviluppo è stato quindi caratterizzato da un'impronta agile, che ha permesso la risoluzione efficace delle problematiche derivanti da cambiamenti concettuali nella comprensione del dominio, espansione di requisiti funzionali o analitici, e più in generale l'ampia gamma di problematiche che emergono durante lo sviluppo di un qualsiasi software.

Il framework si divide in un sistema di controllo principale realizzato in Scala, che si occupa della gestione *end-to-end* degli esperimenti, trasformando l'input dello sperimentatore in risultati e grafici analizzabili, ed un sottosistema scritto in Lua che, orbitando intorno ad un singolo controller ARGoS realizzato secondo l'interfaccia in listato 2.2, espande e modularizza il dominio delle reti booleane lato controller, permettendo la scrittura di sotto-controller che sfruttino il concetto di sussunzione per avere il maggior controllo possibile sui componenti preesistenti e su quelli realizzati, oltre che accomodare tutte le possibili esigenze dello sviluppatore tramite la creazione di librerie apposite per la gestione del dominio.

2.2.1 Design

Il software è stato modellato seguendo la falsariga del modello sperimentale scientifico: un'ipotesi viene convertita in risultato che ne dimostri o meno la validità attraversando una serie di macro-fasi sequenziali, gli input delle quali si considerano essere gli output delle frasi precedenti. Il flusso di controllo che ne

deriva risulta essere facilmente divisibile come in figura 2.4. Le responsabilità funzionali delle varie fasi sono individuate come segue:

- **Inizializzazione -** Creazione ambiente di lavoro, internalizzazione in modello degli input forniti
- **Simulazione -** Consegna del flusso di controllo ad ARGoS, salvataggio ed internalizzazione in modello dei risultati ottenuti
- **Analisi -** Raffinazione dei risultati grezzi tramite calcoli a posteriori e aggregazione
- **Estrazione -** Estrazione e salvataggio dei singoli valori rilevanti allo studio
- **Plotting -** Graficazione dei risultati tramite immagini

Sulla base della compartimentizzazione di queste fasi, esistono due modalità di esecuzione dell'intero codice: una modalità completamente *end-to-end*, in cui si forniscono gli input necessari allo svolgimento dei calcoli e si ottengono direttamente i grafici al termine dell'esecuzione, e una modalità definita come *parziale*, per la quale le fasi di simulazione, analisi e plotting possono essere effettuate singolarmente, sulla base di risultati già raggiunti in precedenza.

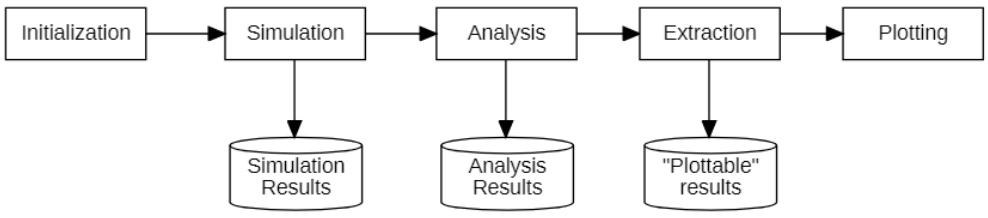


Figura 2.4: Il flusso principale dell'applicazione sperimentale

Il codice che ne risulta, per quanto riassunto all'essenziale per essere adattato ad una trattazione su carta, rispecchia fortemente questa divisione, aderendo nei limiti del tecnicamente possibile al paradigma puro funzionale scelto con l'utilizzo di un linguaggio come Scala. I principali limiti tecnici risiedono nella dimensione in memoria dei valori generati dalle varie fasi: mantenerli costantemente in memoria per raffinarli ed analizzarli non è stato fisicamente possibile, dovendo ripiegare sul salvataggio su disco come contenitore dei valori.

```
object App {
  def main(args: Array[String]): Unit = {
    val settings = Initialization.run(args.toIndexedSeq)
    val simulationSettings = settings.finalizeSettingsForSimulation()
    val analysisSettings =
        settings.finalizeSettingsForContinuousAnalysis()

    val rawValues = Simulation.run(simulationSettings)
    Disk.deflateSimulation(rawValues)

    val results = Analysis.run(analysisSettings)
    Disk.deflateAnalysis(results)

    Plotting.run(analysisSettings)
  }
}
```

Listato 2.3: Lo scheletro dell'applicazione, nella forma *end-to-end* del calcolo

Il listato 2.3 può essere semplicemente letto come

$$E = Plotting(Extraction(Analysis(Simulation(Initialization(input)))))$$

dove E si intende essere la completa esperienza sperimentale da valutare secondo criteri analitici umani.

Nel seguito, una trattazione completa delle responsabilità di ogni fase concettuale del framework, con una descrizione del modello interno e degli artefatti creati o utilizzati durante il calcolo.

Inizializzazione

La fase di inizializzazione prevede di traslare il contenuto di un file di configurazione, attraverso il quale l'utente definisce le proprie modalità sperimentali, all'interno del sistema. Tale internalizzazione viene effettuata attraverso una copia in memoria di tutte le informazioni rilevanti in una struttura definita `AppSettings`, un *Data Transfer Object* interno al sistema incaricato di rendere accessibili ad ogni fase le informazioni peculiari dell'attività che si sta svolgendo. Inoltre, vengono trasportate le informazioni relative a qualsiasi percorso di sistema inerente l'esecuzione, già noto o generato durante la fase stessa.

La creazione di un'istanza di `AppSettings` avviene in due fasi, durante la prima delle quali il sistema determina in che modalità di esecuzione si trovi (parziale o *end-to-end*), generando un ambiente di lavoro che ne rispecchi le necessità; nella seconda fase, una volta fermate le condizioni di lavoro, ogni campo viene riempito con le informazioni necessarie allo svolgimento del calcolo. Tali informazioni vengono distribuite nelle *case class* `AppCustomization`, `AppFolders` e `AppFiles`, e richiamate secondo bisogno tramite un singolo riferimento. In particolare, le informazioni di rilievo dal listato 2.4 vengono trasferite in `AppConfiguration`, il quale contiene un equivalente numero di

campi per permettere un recupero semanticamente esplicito delle informazioni di interesse.

```
{
 "simulation": {
  "seed": 0,
  "distinct-runs": 100,
  "epochs-per-run": 50,
  "epoch-duration": 2500,
  "sensorimotor-frequency": 10,
 },
 "result": {
  "mode": "parallel",
  "result-granularity": "complete",
  "state-trajectory": true
 },
 "empowerment": {
  "steps": 1,
  "convergence-threshold": 1,
  "cutoff": 10
 },
 "robot": {
  "implementation": "environment_obstacle_benchmark.lua",
 },
 "boolean-network": {
  "topology": {
  "nodes-per-network": 100,
  "inputs-per-node": 3,
  "inputs-per-network": 12,
  "outputs-per-network": 2,
  "bias": 0.79,
  "override-output-nodes-bias": true
 },
 "adaptation-parameters": {
  "max-network-input-swaps": 3,
  }
 }
}
```

Nel dettaglio, tramite l'oggetto json *simulation* si specificano:

- **seed** - il seed specifico con il quale inizializzare il generatore di numeri casuali. 0 significa che il sistema è libero di scegliere un seed qualsiasi;
- **distinct-runs** - il numero di repliche diverse di un singolo stesso esperimento;
- **epochs-per-run** - il numero di epoche in una singola replica di esperimento;
- **epoch-duration** - il numero di passi che compone una singola epoca;
- **sensorimotor-frequency** - frequenza percettiva del sistema robotico.

In *result* si specificano una serie di valori relativi alla forma del risultato finale, e alle modalità con le quali questo viene calcolato. In particolare:

- **mode** - specifica la modalità di parallelizzazione del flusso di controllo in simulazione.
- **result-granularity** - specifica la frequenza di aggiornamento da parte del singolo simulatore.
- **state-trajectory** - specifica la volontà di voler tenere traccia dell'intera traiettoria degli stati interni della rete booleana di controllo.

In *empowerment* si specificano i valori per inizializzare tale funzione nel momento in cui la si utilizzi come funzione obiettivo all'interno delle simulazioni. In *robot* si specificano parametri attraverso il quale calibrare vari aspetti sensoriali del robot istanziato, il cui sotto-controller si considera quella specificato tramite il parametro *implementation*. Infine, nei due oggetti che compongono *boolean-network* si specificano i parametri di costruzione ed adattamento delle reti booleane che andranno a costituire i controller dei robot simulati.

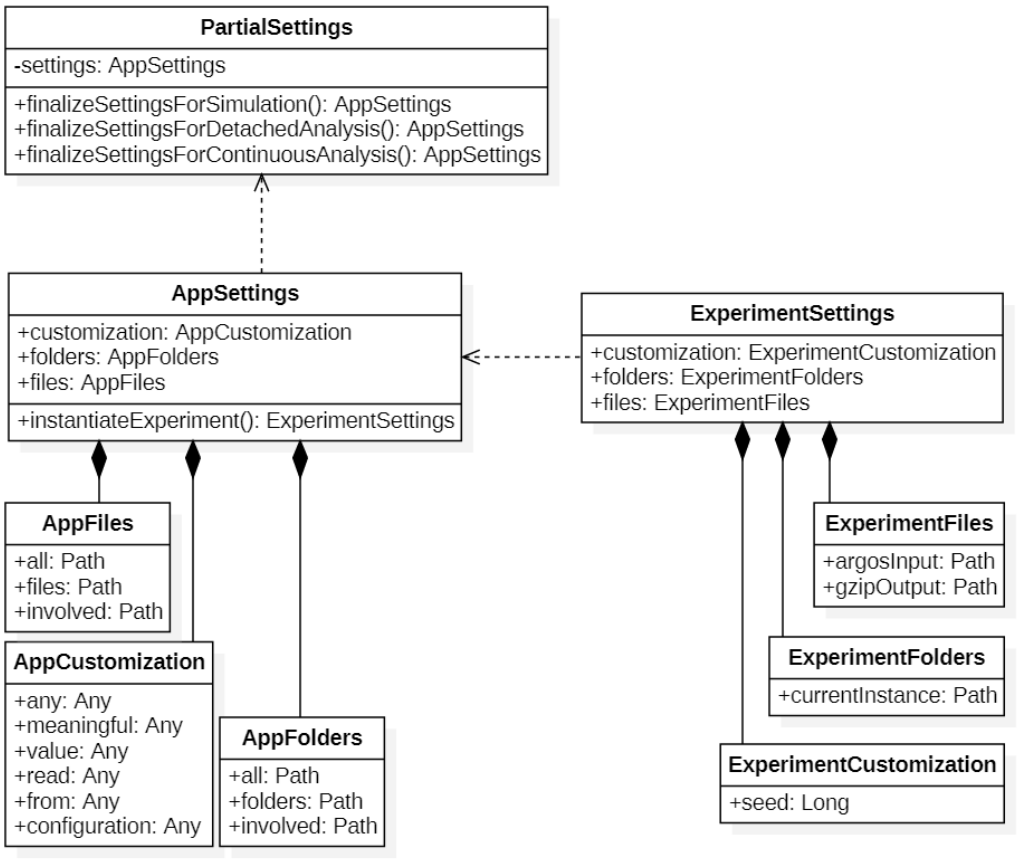


Figura 2.5: Il modello del sotto-sistema di configurazione interna del lato Scala del sistema

Simulazione

Durante la fase di simulazione il controllo del sistema Scala si limita alla generazione delle singole istanze di `ExperimentSettings` (figura 2.5), costruite sulla base della modalità operativa in cui ci si trova, all'accoppiamento di queste con i thread istanziati per gestire la singola istanza di simulazione e alla messa in esecuzione di questi ultimi. Dentro ogni thread, il flusso di controllo viene ceduto ad ARGoS, che effettua la simulazione tramite i precetti descritti nella configurazione iniziale e tramite i calcoli interni del sistema. Il

funzionamento completo del sistema lato ARGoS verrà descritto all'interno dell'appendice A. L'esecuzione di ogni processo ARGoS viene demandata al sistema operativo, motivo per il quale la comunicazione fra sistema ARGoS e Scala avviene tramite un collegamento di quest'ultimo allo `stdout` del processo istanziato, tramite il quale il sistema Scala legge le righe emesse e le traduce in modello interno per effettuare l'analisi e le salva su disco. Il formato e i precetti di salvataggio sono esposti in sezione 2.2.2

Analisi

Nella fase di analisi i risultati vengono inizialmente trasformati in modello interno, esplicitato in figura 2.6. Ogni replica si traduce in una `Adaptation`, che racchiude una lista di `Specimen`, che rappresentano le singole epoche di adattamento. Uno `Specimen` conserva tutte le informazioni relative alla singola epoca: questa classe di elementi viene prima analizzata singolarmente, e successivamente aggregata a livello di `Adaptation` per ottenere dei dati che descrivano l'intera replica nel suo insieme. Infine, tutti gli aggregati vengono salvati in artefatti comuni, attraverso i quali è possibile confrontare gli stessi risultati quantitativi fra diverse repliche.

In particolare, in `Adaptation` viene conservato lo `Specimen` definito come *ancestor*, ovvero la rete booleana generata randomicamente dalla quale è cominciato il processo adattativo.

Uno `Specimen` è composto da:

- **Feature -** una struttura dati contenente il numero dell'epoca di riferimento, il massimo valore della funzione obiettivo raggiunto, il valore raggiunto dalla funzione obiettivo nell'epoca di riferimento, il valore raggiunto dalla funzione *contestant* nell'epoca di riferimento, ovvero

una funzione obiettivo secondaria della quale si desidera comparare le prestazioni.

- **BooleanNetworkState -** la rappresentazione della rete booleana che ha governato l'adattamento del robot in questa specifica epoca: i campi *nodes* e *functions* sono rappresentati internamente da una sequenza e una matrice di `Boolean`; *input* ed *output* da una sequenza di `Double`, *connections* da una matrice di `Double`

- **Traiettoria -** la traiettoria rappresenta l'intera storia dell'epoca di adattamento, "raccontata" tramite la sequenza di stati che la rete booleana ha avuto in ogni *step* di simulazione.

- **Route -** il percorso rappresenta tutte le posizioni assunte dal robot in seguito alle azione del controller. Permette di visualizzare il comportamento della rete booleana in ogni momento della simualzione.

- **Differenziali -** rappresentano il valore della funzione obiettivo al singolo step dell'epoca.

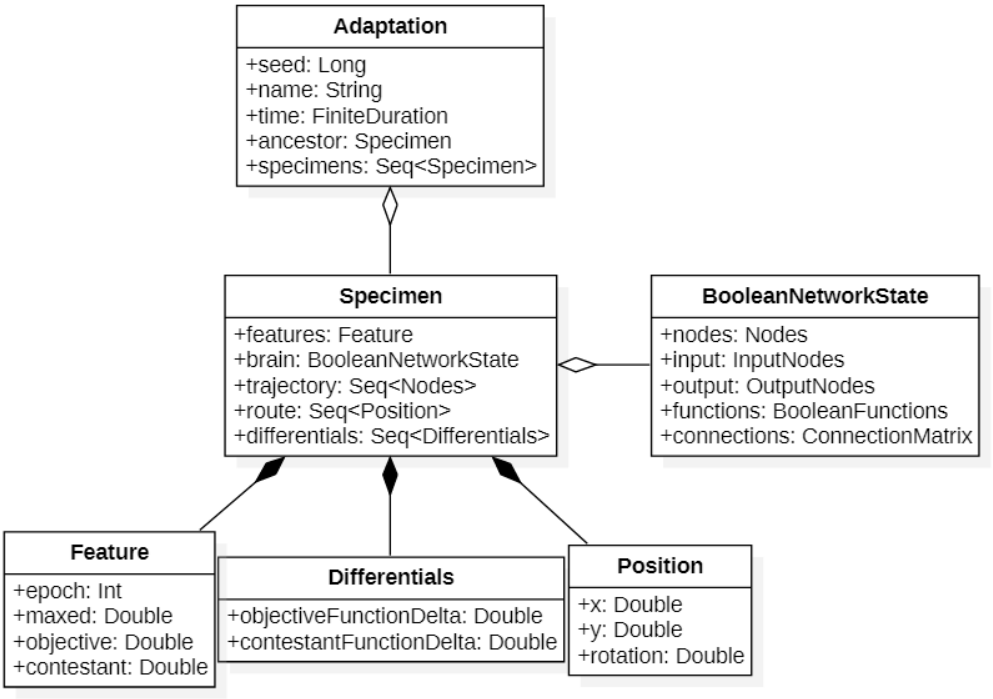


Figura 2.6: Il modello logico interno al lato Scala del sistema

Estrazione

La fase di estrazione prevede di creare una versione dei dati di analisi che rappresentino un parallelo testuale, *human-readable*, dei dati interessanti a livello sperimentale.

Plotting

Il plotting grafica i risultati ottenuti, realizzando una rappresentazione visuale che permetta di mostrare l'andamento degli esperimenti da tutte le possibili angolazioni. I grafici sono raggruppati in tre categorie, perlopiù logiche:

- **Semplici -** i grafici definiti come semplici rappresentano una sola distribuzione di valori nel corso dell'intera replica, e sono perlopiù grafici cartesiani sui quali si diramano i valori delle funzioni.

- **Compositi -** i grafici compositi mostrano due o più degli andamenti degli insiemi di valori generati durante ogni replica, opportunamente combinati perchè sia possibile effettuare un confronto su valori raggiunti e comportamento.

- **Aggregati -** attraverso i grafici aggregati si realizzano viste trasversali sul comportamento generale della replica, mostrando i comportamenti delle distribuzioni di valori all'interno delle varie epoche di una replica piuttosto che dei singoli valori in sè. Sono concretizzati perlopiù attraverso dei diagrammi a scatola e baffi.

2.2.2 Formato di comunicazione intraprocesso

Per effettuare la comunicazione intraprocesso fra il framework ed ARGoS è stato necessario sviluppare un formato di comunicazione apposito che permettesse di trasferire in maniera efficace la massiva mole di informazioni generate dal simulatore, oltre che a permetterne il salvataggio su disco.

Per quantificare tale mole, si consideri un singolo esperimento di simulazione lanciato con le seguenti caratteristiche:

- 100 repliche distinte

- 50 epoche di adattamento per replica

- 2500 passi di simulazione per epoca

- una rete booleana inizializzata ad avere 100 nodi per rappresentare lo stato interno, con 3 connessioni fra i nodi, 12 nodi di input e 2 di output

- il calcolo di due differenti funzioni obiettivo

Tale configurazione porta alla generazione, nella singola epoca, di:

- un array booleano da 100 elementi per ospitare lo stato interno della rete booleana;
- 3 array numerici da 12, 2 e 4 elementi, per ospitare rispettivamente gli indici dei nodi di input, output e i valori delle funzioni obiettivo calcolate, nelle versione attuale e massimizzata durante l'esecuzione;
- una matrice booleana con dimensioni $8x100$, per rappresentare le funzioni booleane che guidano la rete booleana;
- una matrice numerica con dimensione $3x100$, per contenere le connessioni fra i singoli nodi interni di una rete booleana;
- una matrice booleana con dimensioni $100x2500$, per contenere la traiettoria durante l'epoca degli stati interni della rete booleana;
- una matrice booleana con dimensioni $2x2500$ per contenere i differenziali delle funzioni obiettivo calcolate nella singola epoca;
- una matrice numerica con dimensioni $3x2500$, per contenere le informazioni relative al percorso effettuato dal robot nella singola epoca.

La comunicazione della totalità di questi dati si considera effettuata al completamento di ogni epoca di ogni replica dal sistema ARGoS al sistema Scala, ed al termine di ogni replica salvata su disco dal sistema Scala: tale mole di dati va considerata moltiplicata per 50, per esprimere effettivamente una singola replica, per 100, per considerare tutte le repliche di un singolo esperimento e per 36, tante sono complessivamente gli esperimenti effettuati per realizzare questo lavoro di tesi.

L'ipotesi di utilizzare un formato *mainstream* come il Json si è dimostrata immediatamente impraticabile, a causa dell'eccessiva verbosità del linguaggio: tale verbosità è stato un problema già dimostrato in precedenti lavori di tesi che seguivano le stesse modalità. In questa particolare iterazione il problema è stato esacerbato dalla maggiore quantità di informazioni che si è cercato di estrarre dal simulatore.

Il listato 2.5 esprime in maniera evidente il problema: si stanno trasportando due epoche, ognuna delle quali è costituita da un valore numerico, un array numerico da 3 elementi, un array booleano da 4 elementi, una matrice numerica di dimensione $3x3$ e una matrice booleana di dimensione $3x3$. L'overhead del formato rispetto alla scarsa informazione trasportata è evidente, anche utilizzando meccanismi per ridurre il peso dei dati[1] come utilizzare nomi non significativi e usare valori binari al posto dei lessemi *true* e *false*.

[1]Non utilizzate nel listato d'esempio per garantire leggibilità nella trattazione.

```
{
 "1" : {
  "value": 12345,
  "numeric-array": [1, 2, 3],
  "boolean-array": [true, false, true, false],
  "numeric-matrix": [
   [1, 2, 3],
   [4, 5, 6],
   [7, 8, 9]
  ],
  "boolean-matrix": [
   [true, false, true],
   [true, false, true],
   [true, false, true]
  ]
 },
 "2": {
  "value": 67890,
  "numeric-array": [4, 5, 6],
  "boolean-array": [true, false, true, false],
  "numeric-matrix": [
   [7, 8, 9],
   [4, 5, 6],
   [1, 2, 3]
  ],
  "boolean-matrix": [
   [true, false, true],
   [true, false, true],
   [true, false, true]
  ]
 }
}
```

Listato 2.5: Struttura ipotetica di un file di salvataggio risultati concretizzato attraverso il formato json

Estrapolando le necessità richieste dal formato, risulta necessario esclusivamente trasportare una moltitudine di array e matrici, di tipo numerico e booleano (il caso dei valori singoli può essere ridotto al caso di array formati da un unico elemento) associati in maniera inscindibile ad un'epoca di riferimento nella maniera il più succinta possibile. Tali requisiti vengono soddisfatti dal formato definito, chiamato `N` e presentato nel listato 2.6, che trasporta le stesse informazioni del listato 2.5 in maniera estremamente compressa. Le regole di trasformazione si delineano come segue:

- il contenuto informativo di una singola epoca si considera posizionato su una singola riga;
- l'associazione ordinale ad un'epoca si considera relativo all'ordine interno delle righe del file;
- ogni elemento trasferito si considera associato ad una lettera che ne permetta l'individuazione ed il recupero. Non esistono altre lettere all'interno del file;
- gli elementi di un array numerico si considerano separati da una virgola;
- gli elementi di un array booleano non si considerano separati da nulla, e i valori *true* e *false* sono ovviamente traslitterati nei corrispettivi valori binari 1 e 0;
- i diversi array che costituiscono una matrice sono separati dal simbolo £

```
v12345n1,2,3b1010N1,2,3£4,5,6£7,8,9B101£101£101
v67890n4,5,6b1010N9,8,7£6,5,4£3,2,1B101£101£101
```

Listato 2.6: Struttura di un file di salvataggio risultati concretizzato attraverso il formato N

Considerando i caratteri utilizzati per rappresentare le informazioni nei due formati, riducendo i nomi delle chiavi Json ad un solo carattere e sostituendo 1 e 0 a *true* e *false*, e senza considerare tutti i caratteri bianchi, si ha comunque un risparmio del 50% di spazio in memoria e su disco con l'adozione di `N`, che utilizza solo 94 caratteri per trasferire le informazioni, contro i 205 del Json.

Capitolo 3

Esperimenti

Dopo una breve introduzione mirata ad elencare le proprietà di quelli che sono considerati essere gli esperimenti classici nel campo dell'adattamento online di reti booleane, nel seguente capitolo si vanno ad espandere i precetti con i quali si è arrivati a comporre gli esperimenti che hanno dato vita alla tesi. In particolare, nella sezione *Processo sperimentale* si sviluppa attraverso quali esperimenti si siano raffinate le conoscenze necessarie ad effettuare la sperimentazione obiettivo; successivamente, nella stessa sezione, viene definita la logica comune attraverso la quale si sono strutturate le simulazioni; infine, nelle sezioni *Navigazione fra ostacoli*, *Ricerca spaziale assistita* e *Navigazione su tracciato* vengono presentati gli esperimenti effettuati, espressi in funzione del loro ambiente e della loro configurazione iniziale, seguiti ovviamente dalle principali conclusioni date dai risultati quantitativi.

3.1 Esperimenti classici

Data la struttura e le peculiarità di una rete booleana, sono stati selezionati degli esperimenti in grado di esprimere un comportamento significativo appog-

giandosi su un tipo di controllo basato esclusivamente sui valori delle funzioni matematiche interne, e per i quali sia possibile determinare un indice della correttezza in maniera non ambigua. Inoltre, data la complessità computazionale del calcolo dell'empowerment è stato necessario ricadere in una gamma di esperimenti parecchio semplici, che non dipendano dall'interconnessione di molti sensori per essere svolti.

Navigazione fra ostacoli L'esperimento di navigazione fra ostacoli prevede la massimizzazione del movimento in un ambiente costellato di ostacoli fissi. Per svolgerlo è necessario il solo uso di un insieme di sensori di prossimità.

Ricerca spaziale assistita L'esperimento di ricerca spaziale assistita utilizza la fototassi come strumento per guidare il robot ad individuare una zona di interesse all'interno dell'ambiente. Per svolgerlo vengono utilizzati un insieme di sensori di luminosità e di terreno.

Navigazione su tracciato L'esperimento di navigazione su terreno prevede la massima aderenza ad un percorso tracciato sul terreno, utilizzando soltanto un insieme di sensori di pavimento.

3.2 Processo sperimentale

L'obiettivo degli esperimenti è quantificare e valutare qualitativamente l'impatto dell'utilizzo dell'empowerment nel processo di apprendimento di una RBN in un contesto in cui l'ambiente in cui questa è situata si trovi a cambiare con dinamiche non conosciute nè prevedibili dall'*automaton*. In particolare, si vuole :

1. valutare analiticamente i benefici dell'applicazione del solo *empowerment* come motore del processo di adattamento;

2. determinare se l'applicazione del concetto di empowerment sia in grado di migliorare in termini qualitativi i risultati ottenuti dall'automaton;

3. determinare come l'applicazione del concetto di empowerment influisca sul processo di adattamento nel momento di generare un esemplare in grado di affrontare il nuovo ambiente;

La dinamicità dell'ambiente si considera proporzionalmente limitata sia dalle capacità computazionali dei processori a disposizione, che dalle capacità del simulatore in cui tale ambiente viene generato, che dall'ovvia relazione che intercorre fra i due concetti: la potenza di calcolo disponibile non è stata potuta essere destinata a simulare un ambiente ricco e dinamico e contemporaneamente applicare algoritmi complessi, in quanto non sarebbe stato possibile affrontare il costo dell'operazione in termini di tempo necessario al compimento del calcolo.
Tali limitazioni sono state aggirate riformulando il concetto di mutevolezza di ambiente. La dinamicità di uno spazio altri non è che il susseguirsi di una serie di percezioni di stati: volendo analizzare il comportamento di un robot in un ambiente soggetto a perturbazioni, si è deciso quindi di creare due ambienti per ogni esperimento, in maniera tale da rappresentare lo stato spaziale nella condizione di equilibrio precedente alla perturbazione (o ambiente A) e quello immediatamente successivo al nuovo equilibrio ristabilito (o ambiente B), in questo caso senza indagare né la natura né la durata del fenomeno.
Al robot controllato via RBN viene quindi fatta effettuare una sessione di adattamento progressivo all'interno dell'ambiente A, al termine della quale viene estratto l'esemplare più capace secondo i precetti dell'esperimento in questione; questo viene successivamente iniettato come progenitore nell'ambiente B,

effettuando una seconda fase di adattamento. Rimuovendo dalla simulazione il concetto di fenomeno prolungato nel tempo e soffermandosi sugli istanti temporali immediatamente precedenti al suo avvenimento ed immediatamente successivi al ripristino dell'equilibrio, è stato possibile eseguire l'esperimento tramite due simulazioni ad ambiente statico, la cui concatenazione delle quali e dei risultati ottenuti può essere considerata a tutti gli effetti come un continuo temporale.

Benchmark singola arena Per ogni arena accoppiata attraverso l'esperimento è stata effettuata una sessione di adattamento singola, in maniera tale da determinare quale fosse il comportamento registrato in una delle sottoparti del'esperimento complessivo. All'interno delle simulazioni di benchmark ad arena singola viene utilizzata una funzione obiettivo, attraverso la massimizzazione della quale viene portato avanti l'adattamento del robot. Il risultato delle simulazioni viene espresso attraverso un'istogramma che rappresenta, per ogni epoca, il massimo valore raggiunto dalla funzione obiettivo, un *box-plot* che esprime il cambiamento di distribuzione dei valori massimi della funzione obiettivo fra la prima epoca della simulazione e l'ultima, e una serie temporale definita come RLD (*Run length distribution*) che mostra, proiettata per ogni epoca, la percentuale di repliche che hanno superato una data percentuale della funzione obiettivo massima raggiunta, ovvero l'aderenza al task per il quale si è modellata il processo euristico.

Tuning singola arena Il processo di tuning è una fase a simulazione singola in cui l'empowerment viene utilizzato come funzione obiettivo attraverso il calcolo della quale si guida il processo di adattamento del robot. L'obiettivo è duplice: individuare un iperparametro di configurazione da applicare nella

fase di simulazione concreta, in maniera tale da massimizzare i benefici dell'empowerment riducendo contemporaneamente al minimo gli effetti deleteri sul tempo di simulazione; individuare possibili correlazioni fra valori di empowerment e task specifici, per determinare in che contesti la sola applicazione del concetto possa risultare migliore di una funzione obiettivo determinata *in vitro*. Inoltre, effettuandosi sulla singola arena, è possibile confrontare i risultati con la fase di benchmark appena descritta. Il parametro di configurazione ricercato è definito *cut-off*. Si individua effettuando un'analisi a posteriori di un intero processo di *tuning*, e rappresenta il numero mediano di epoche necessario affinchè l'empowerment si possa considerare massimizzato.

Benchmark sperimentale Per ogni tipo di esperimento è stato necessario fissare quali fossero le prestazioni originali del sistema, sulla base delle quali poter valutare i risultati dell'applicazione dell'empowerment al processo adattativo in un contesto in cui avvengono modifiche all'ambiente originale. Questa fase del benchmark prevede l'applicazione del processo sperimentale descritto precedentemente, in cui la miglior rete viene iniettata come controller del progenitore della seconda arena. Le coppie di simulazioni di benchmark si intendono sempre configurate con gli stessi valori e precetti degli ambienti sui quali si andrà ad effettuare il test dell'ipotesi, ma l'evoluzione del robot viene guidata attraverso funzioni obiettivo mirate allo svolgimento dello specifico task.

Sperimentazione La fase di sperimentazione rappresenta la coppia di simulazioni durante la quale l'empowerment viene utilizzato in un contesto applicativo per verificarne le proprietà ipotizzate. Nell'ambiente A esiste una fase di pre-adattamento del processo dalla durata determinata dai risultati ottenuti durante la simulazione di tuning appena descritta, durante la quale il livello di

empowerment viene massimizzato. Al termine di questa fase inizia il processo di adattamento basato su euristica, basandosi non più su una RBN ma su una EBN, *Empowered Boolean Network*. Al termine della simulazione nell'ambiente A l'automaton viene trasferito nell'ambiente B, senza che in questa seconda fase si effettui il processo di pre-adattamento.

La funzione obiettivo mirata allo svolgimento dello specifico task si considera la stessa per tutte le fasi appena descritte, nel contesto del singolo esperimento.

3.3 Analisi

I risultati di ogni replica della simulazione attraversano un'analisi in due fasi, la prima delle quali di tipo quantitativa, la seconda qualitativa. Per ognuna vengono estrapolate le caratteristiche del miglior esemplare prodotto. Da ogni replica vengono inoltre estratti e raggruppati in artefatti appositi:

- i valori delle funzioni obiettivo massimizzate;
- i valori delle funzioni obiettivo calcolata ad ogni epoca;
- i valori della funzione *contestant* (vedi sezione 2.2) calcolata ad ogni epoca;
- le posizioni occupate dal robot in ogni fase della replica

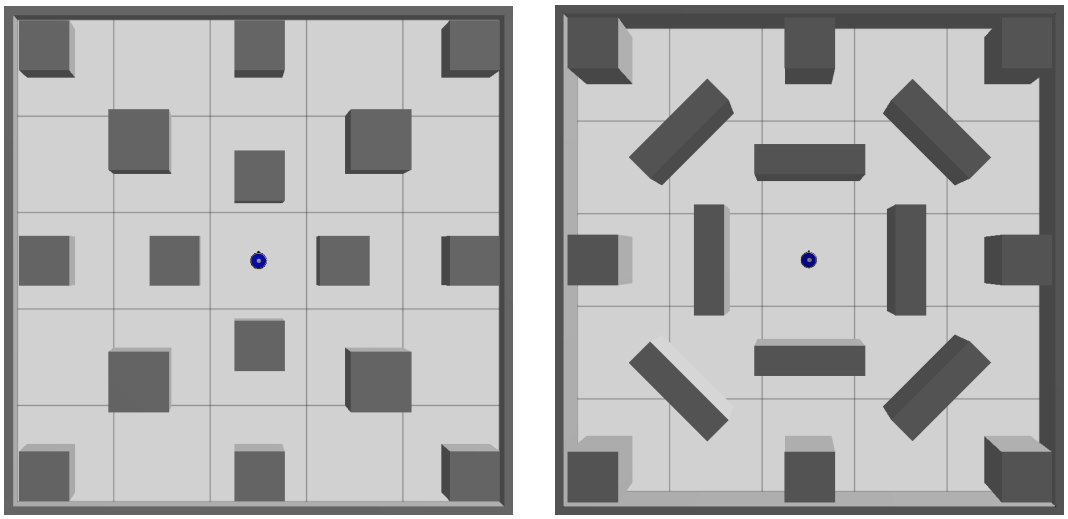
Nella fase di analisi qualitativa vengono effettuati confronti fra fasi diverse dell'esperimento, misurando in maniera concreta le prestazioni raggiunte. I risultati vengono mostrati in dei diagrammi a scatola e baffi che mostrano la progressione delle distribuzioni dei valori nelle fasi di *benchmark* e simulazione sperimentale in più momenti dell'adattamento all'interno della seconda nicchia ambientale dell'esperimento, e tramite degli RLD si confrontano le percentuali

di esemplari di successo prodotti nelle due fasi. Infine, viene analizzata la validità statistica dei risultati raggiunti tramite il test di Wilcoxon e il test di Mann-whitney.

3.4 Navigazione fra ostacoli

L'esperimento di navigazione fra ostacoli prevede di massimizzare la quantità di spostamento del robot favorendo il movimento rettilineo, mantenendo al tempo stesso una distanza adeguata dagli ostacoli che incontra.

Setup ambiente Gli ambienti sono costituiti da arene di dimensione 5x5, all'interno delle quali il robot è vincolato da un perimetro che risiede sul margine di tale area. Gli ostacoli sono inamovibili. Il robot si suppone nascere al centro dell'arena, senza che inizialmente tocchi nessun ostacolo.



(a) Ambiente A (b) Ambiente B

Figura 3.1: Coppia di ambienti nei quali è stato effettuato l'esperimento di navigazione

Nella creazione delle due arene si è cercato di mantenere costante la superficie calpestabile: data la superficie totale di $25m^2$, in entrambe si è occupata un'area totale di $5.1m^2$ con ostacoli. La disposizione degli ostacoli nei due esperimenti è stata scelta in maniera tale che da un ambiente all'altro fosse necessario potersi spostare grazie ad un apprendimento puro del concetto di *obstacle avoidance*, e non semplicemente memorizzando meccanicamente un percorso in particolare.

Setup robot Il robot è dotato di dodici sensori di prossimità, situati in maniera equidistante lungo il proprio perimetro. Tali sensori rilevano ostacoli ad una distanza massima di 10 centimetri, e ognuno di loro produce un valore nel range $[0, 1]$, dove 0 indica la mancata percezione di alcunchè e 1 indica che ci si trova in stato di collisione con un ostacolo.

Funzione obiettivo La funzione che il processo euristico di adattamento intende massimizzare consiste in

$$f_{epoch} = \sum_{1}^{t}(1 - P_{max}) * (1 - \sqrt{|L - R|}) * \left(\frac{L + R}{2}\right)$$

dove:

- t rappresenta il numero totale di passi per ogni epoca della simulazione;
- P_{max} il valore del sensore di prossimità con attivazione più alta;
- L (corr. R) rappresenta l'attivazione in valore binario della ruota sinistra (corr. destra), dove il valore 1 che esprime una ruota attiva e 0 una ruota non attiva.

Lo scopo di tale funzione è quello di premiare, nell'intera epoca del processo di adattamento, un comportamento che preveda l'attivazione di entrambe le

ruote (quindi un movimento rettilineo) mantenendosi contemporaneamente il più lontano possibile dagli ostacoli.

Setup simulazione Il simulatore effettua 100 repliche dell' esperimento, ognuna delle quali consiste in 50 epoche di adattamento divise in 2500 *tick* dell'ambiente, che nella rappresentazione interna del tempo logico del simulatore corrispondono a 250 secondi di tempo effettivo. Un'intera replica dell'esperimento si considera quindi equivalere a 3 ore, 28 minuti e 20 secondi di tempo effettivo.

Nel caso delle simulazioni ad ambiente multiplo il numero di repliche si considera raddoppiato, in quanto l'adattamento viene ripetuto per intero sulla seconda arena una volta terminato il primo ciclo.

I parametri del processo di inizializzazione della RBN di controllo sono fissati secondo i parametri standard delle RBN di tipo criticale[6]:

- 100 nodi complessivi;
- 3 connessioni entranti per ogni nodo;
- un *bias* di generazione delle funzioni booleane per ogni nodo pari a 0,79.

Inoltre, gli altri parametri di generazione sono stati fissati come segue:

- 12 nodi di input;
- 2 nodi di output;
- un *bias* di generazione delle funzioni booleane relative all'attivazione dell'output pari a 0.5 [4];
- 3 possibili *rewiring* dei nodi di input nella fase di adattamento all'ambiente.

3.4.1 Benchmark singola arena

Con la simulazione ad arena singola si vogliono misurare le prestazioni di base ottenute dall'adattamento tramite funzione obiettivo, in maniera tale che sia possibile un successivo confronto con forme di adattamento diverse ma situate nello stesso ambiente.

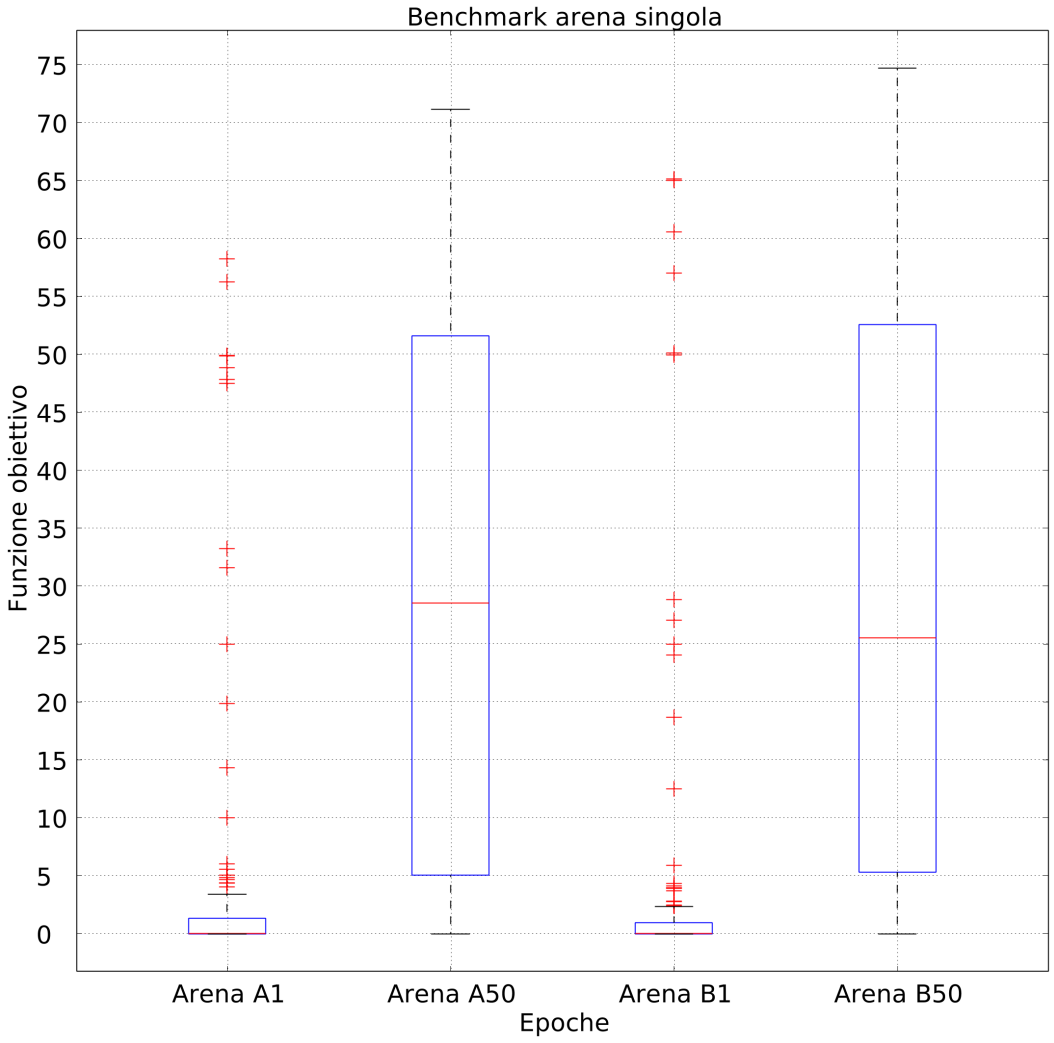
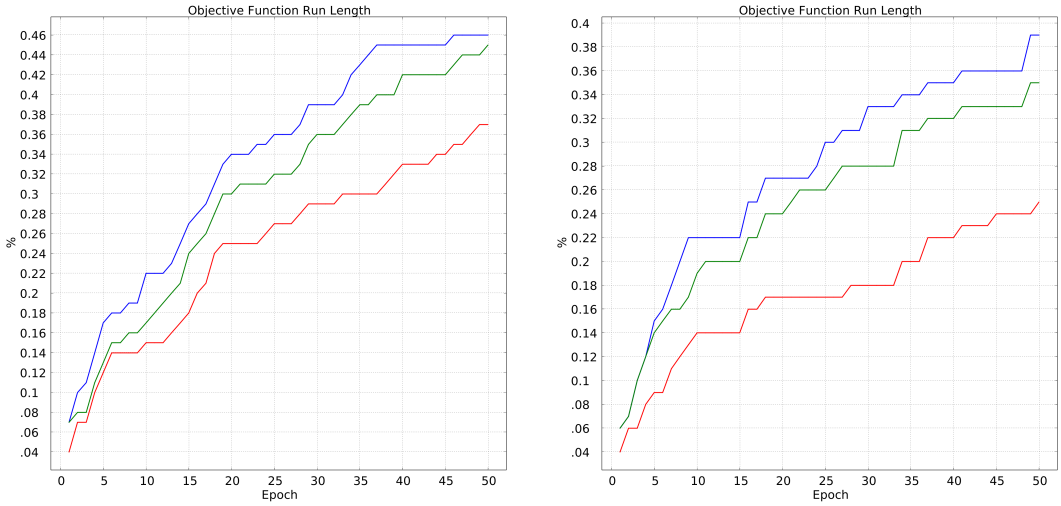


Figura 3.2: Nella metà sinistra (risp. destra), la distribuzione dei valori di funzione obiettivo di benchmark per la sola arena A (risp. B)



(a) RLD su ambiente A

(b) RLD su ambiente B

Figura 3.3: *Run length distribution* degli ambienti di benchmark ad arena singola: le tracce blu, verdi e rosse rappresentano rispettivamente la soglia di calcolo posta al 50, 60 e 70 per cento del valore massimo raggiunto

Tramite i plot in 3.3 viene suggerita una maggiore facilità di esplorazione del primo ambiente, con il 46% delle RBN che superano il 50% del massimo valore raggiunto, e il 37% che superano il 70%. Questo viene contrapposto al 39% di repliche in cui viene oltrepassato il 50%, con solo il 25% che superano il 70% del massimo valore raggiunto.

3.4.2 Tuning singola arena

Sia per l'ambiente A che l'ambiente B si è ottenuto che il valore di *cut-off* sia pari a 2. Questo sarà quindi il valore oltre il quale verrà effettuato il ritorno alla massimizzazione di funzione obiettivo nelle simulazioni effettuate in Sperimentazione.

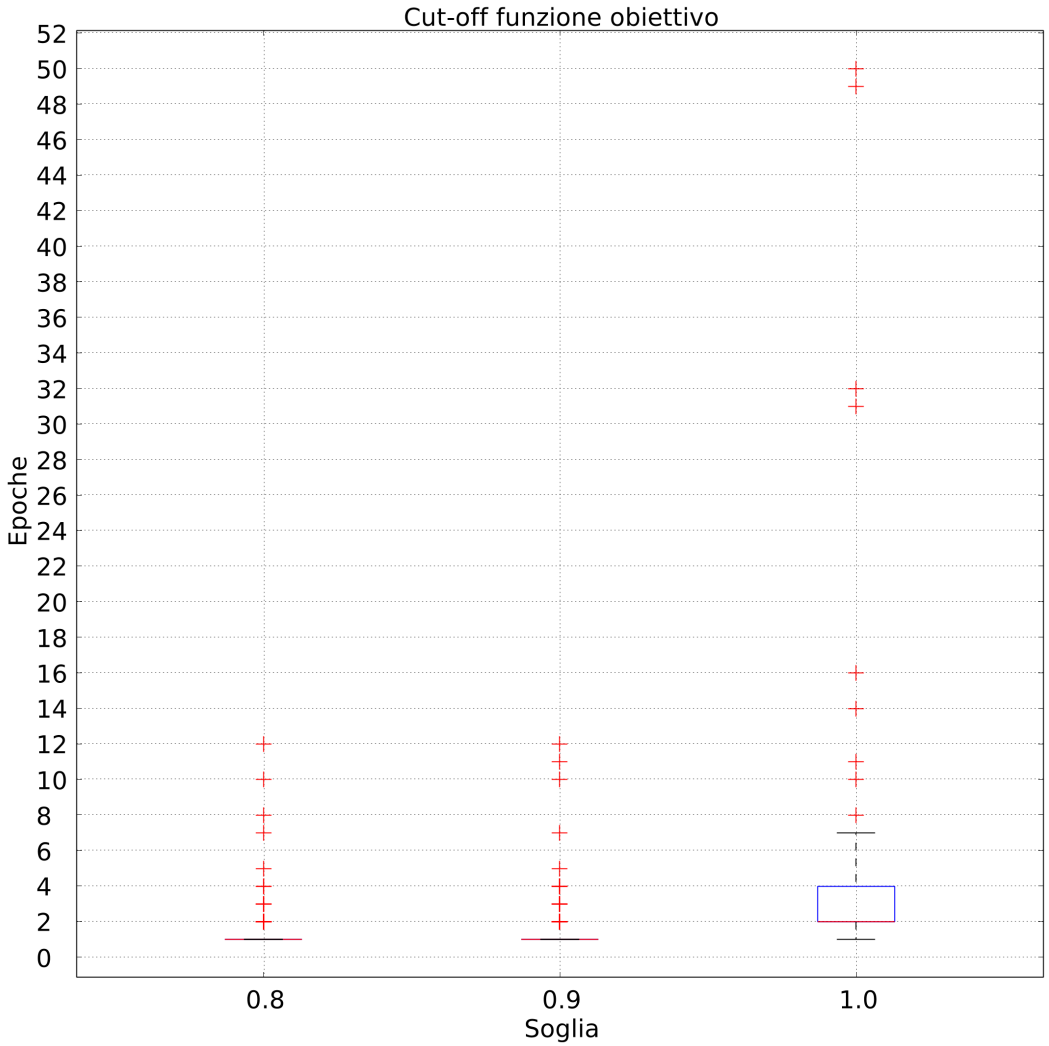


Figura 3.4: Soglie di cutoff su ambiente A

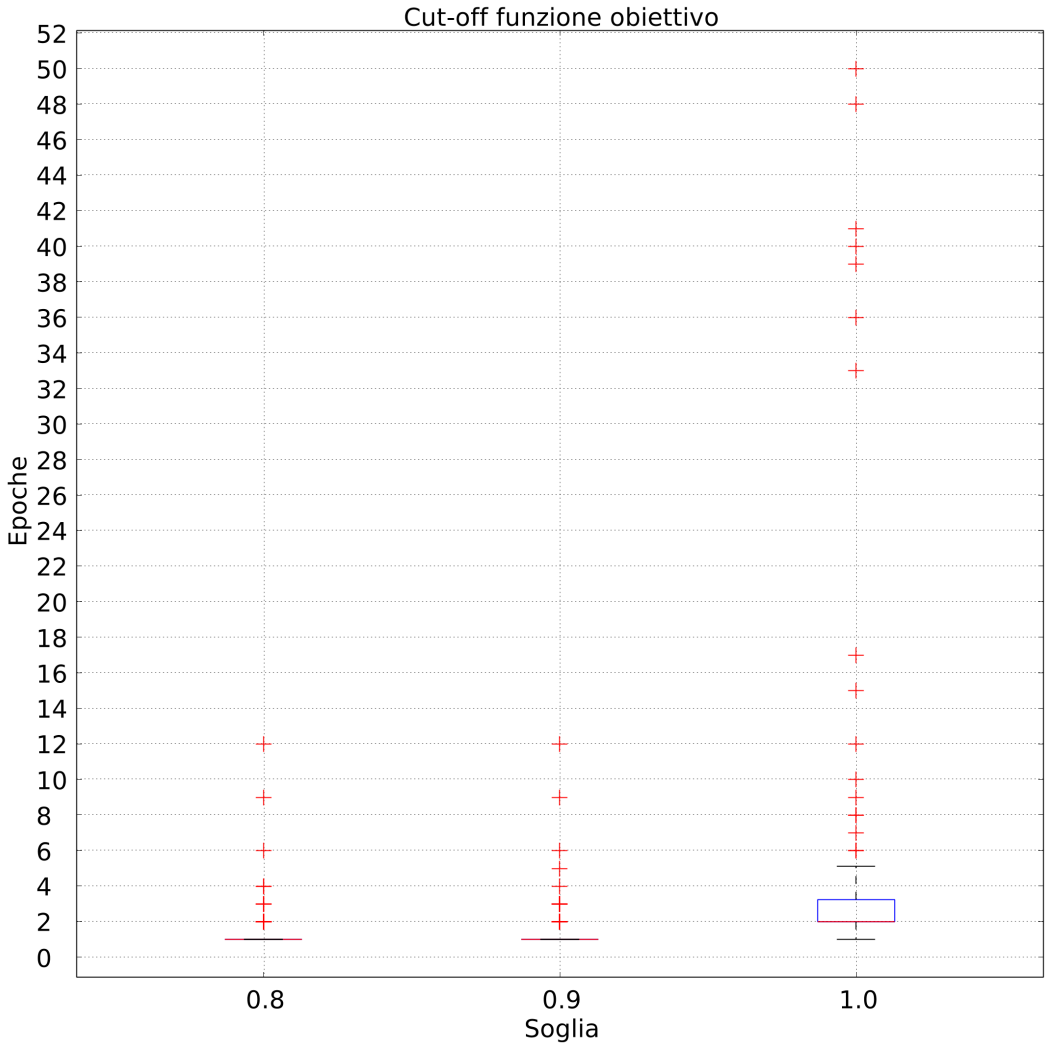


Figura 3.5: Soglie di *cut-off* su ambiente B

Inoltre si è avuta l'opportunità di verificare che sia possibile raggiungere valori adeguati di funzione obiettivo anche senza utilizzare quest'ultima come motore principale dell'adattamento.

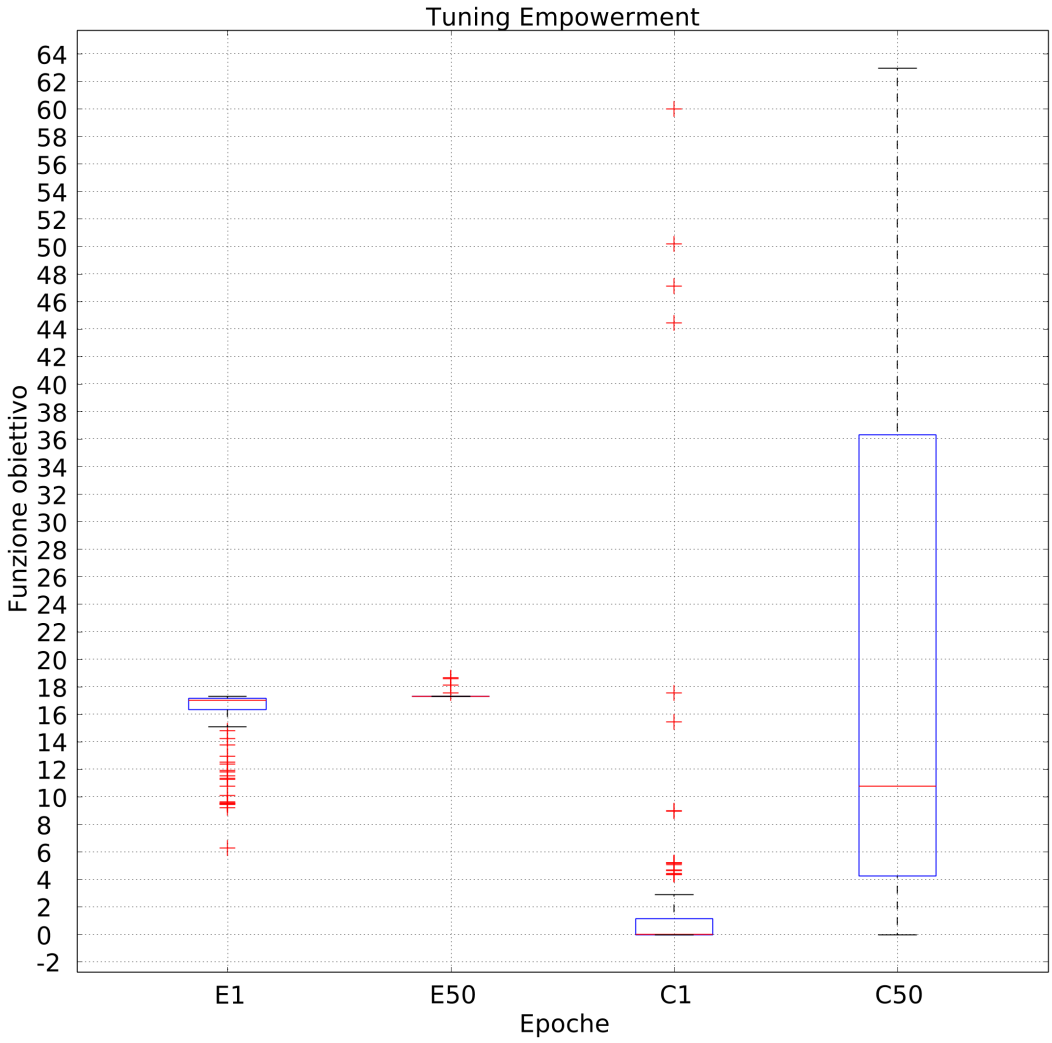


Figura 3.6: Distribuzioni ad epoca 1 e 50 dei valori della funzione di empowerment (sinistra) e corrispondenti valori obiettivo (destra) su ambiente A

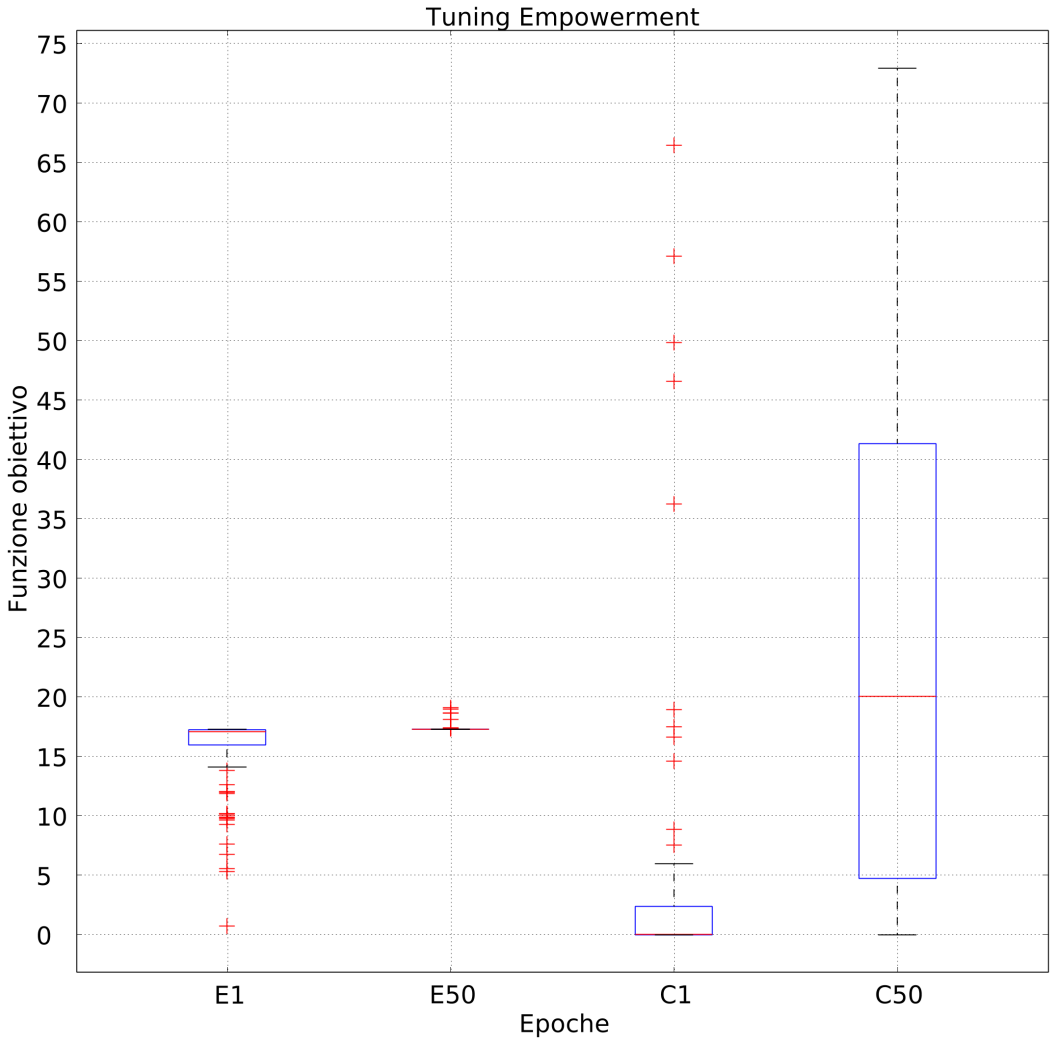


Figura 3.7: Distribuzioni ad epoca 1 e 50 dei valori della funzione di empowerment (sinistra) e corrispondenti valori obiettivo (destra) su ambiente B

3.4.3 Benchmark sperimentale

I risultati della fase di benchmark con trasferimento su ambiente diverso delineano la capacità della RBN di riacquisire l'adattamento raggiunto nella prima fase della simulazione, in maniera indipendente dall'ordine in cui vengono fatti partire i robot. Per quanto il terzo quartile relativo alla distribuzione

di valori nella seconda arena non raggiunga gli stessi valori del proprio corrispettivo della prima, la mediana della distribuzione si è notevolmente alzata nel passaggio alla seconda fase.

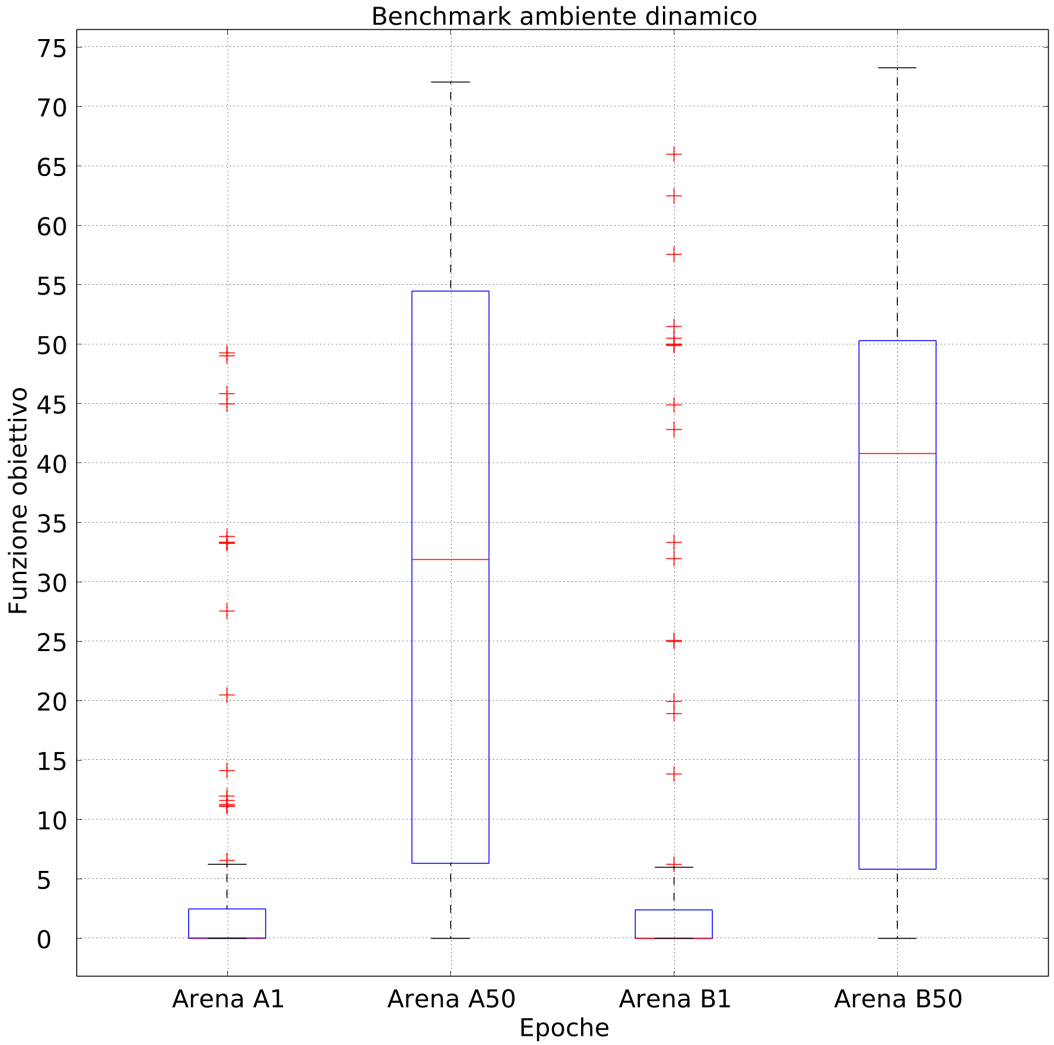


Figura 3.8: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena A, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena A nell'arena B

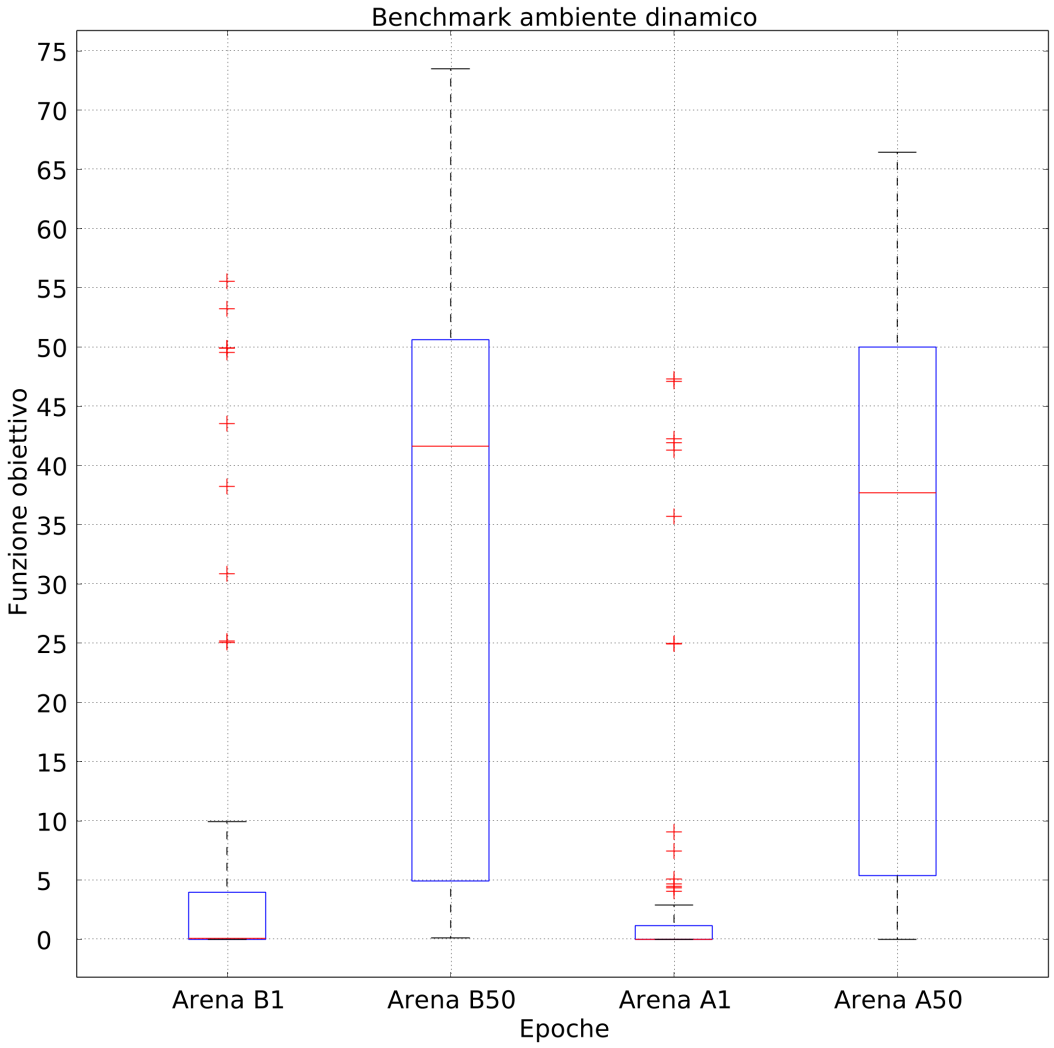


Figura 3.9: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena B, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena B nell'arena A

3.4.4 Sperimentazione

La fase di sperimentazione, eseguita secondo i precetti descritti in sezione 3.2 mostra come in entrambe le simulazioni sia stato possibile riacquisire le capacità sviluppate durante la prima fase, e superarle di netto nella simulazione

in cui si passa dall'ambiente B all'ambiente A.

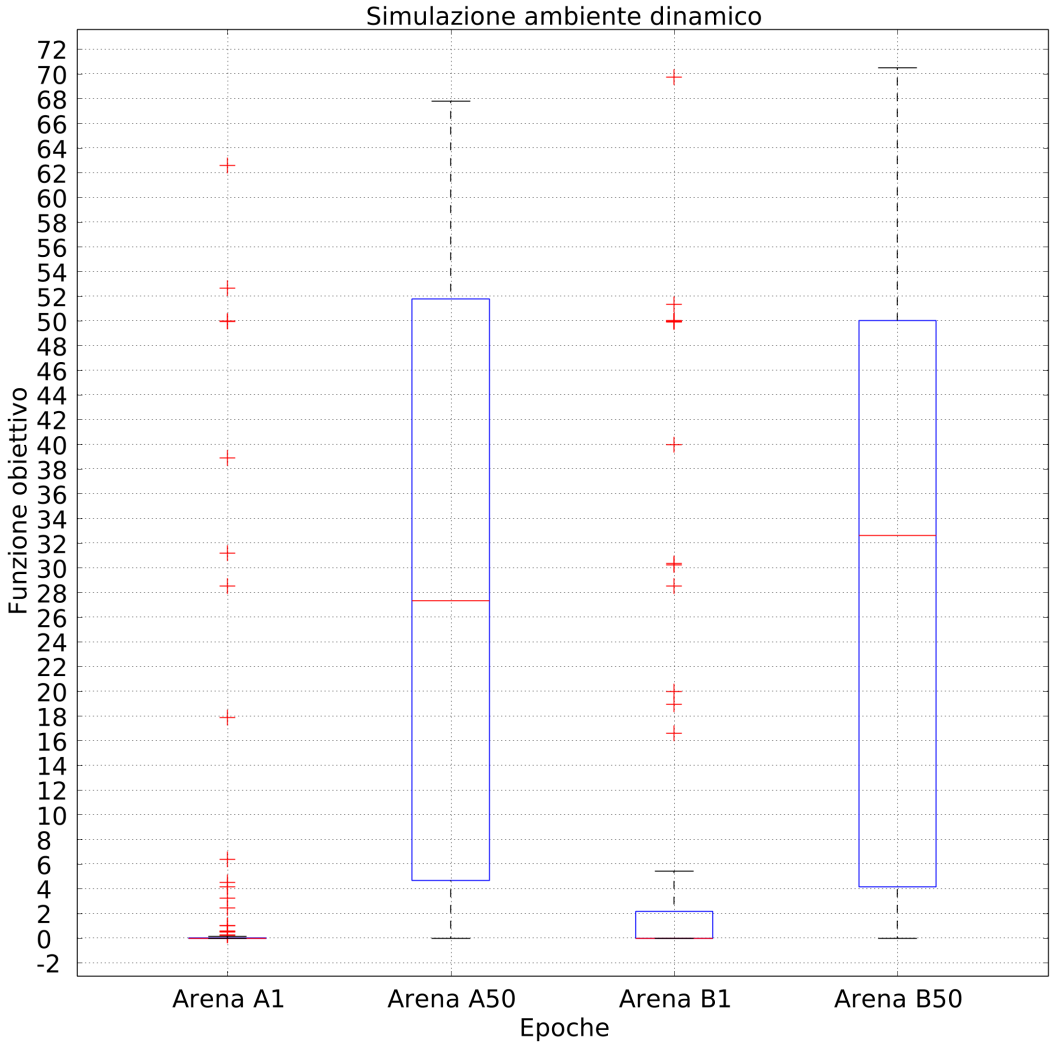


Figura 3.10: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena A, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena B

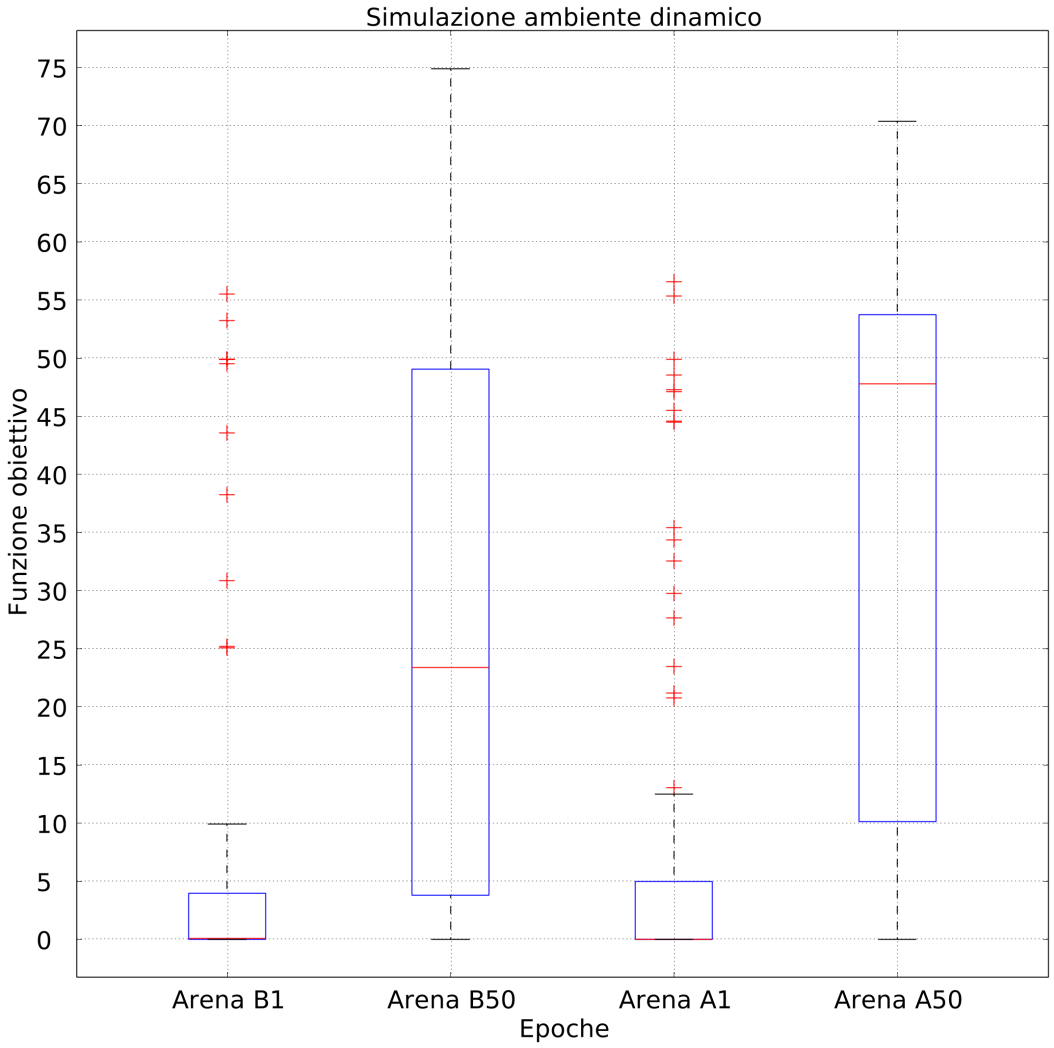


Figura 3.11: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena B, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena A

3.4.5 Risultati

Il confronto fra le distribuzioni dei valori ottenute in benchmark e fase sperimentale nella simulazione AB (pre-adattamento in A con trasferimento in B) mostra come nella fase finale dell'esperimento i risultati ottenuti in benchmark

non riescano a venire eguagliati da quelli ottenuti in fase sperimentale, se non con un ritardo di parecchie epoche. La figura 3.12 mostra come la mediana delle distribuzione di valori ottenuti in benchmark sia sempre superiore, e sempre più velocemente, rispetto alla fase sperimentale.

Questo risultato viene contraddetto dal confronto della simulazione in cui l'ordine delle arene viene invertito, effettuando quindi il trasferimento del migliore esemplare nell'arena A. In questo caso (figura 3.13) esiste una netta superiorità dei valori della distribuzione della fase sperimentale. In particolare, si delinea un incremento velocissimo del terzo quartile nel *boxplot* denominato S10, ovvero i risultati ottenuti all'epoca 10 della fase di simulazione. Per quanto questo fenomeno si sia riscontrato anche nella fase di benchmark (B10), il trasferimento dell'EBN sperimentale sembra aver incrementato la velocità dell'incremento dei valori.

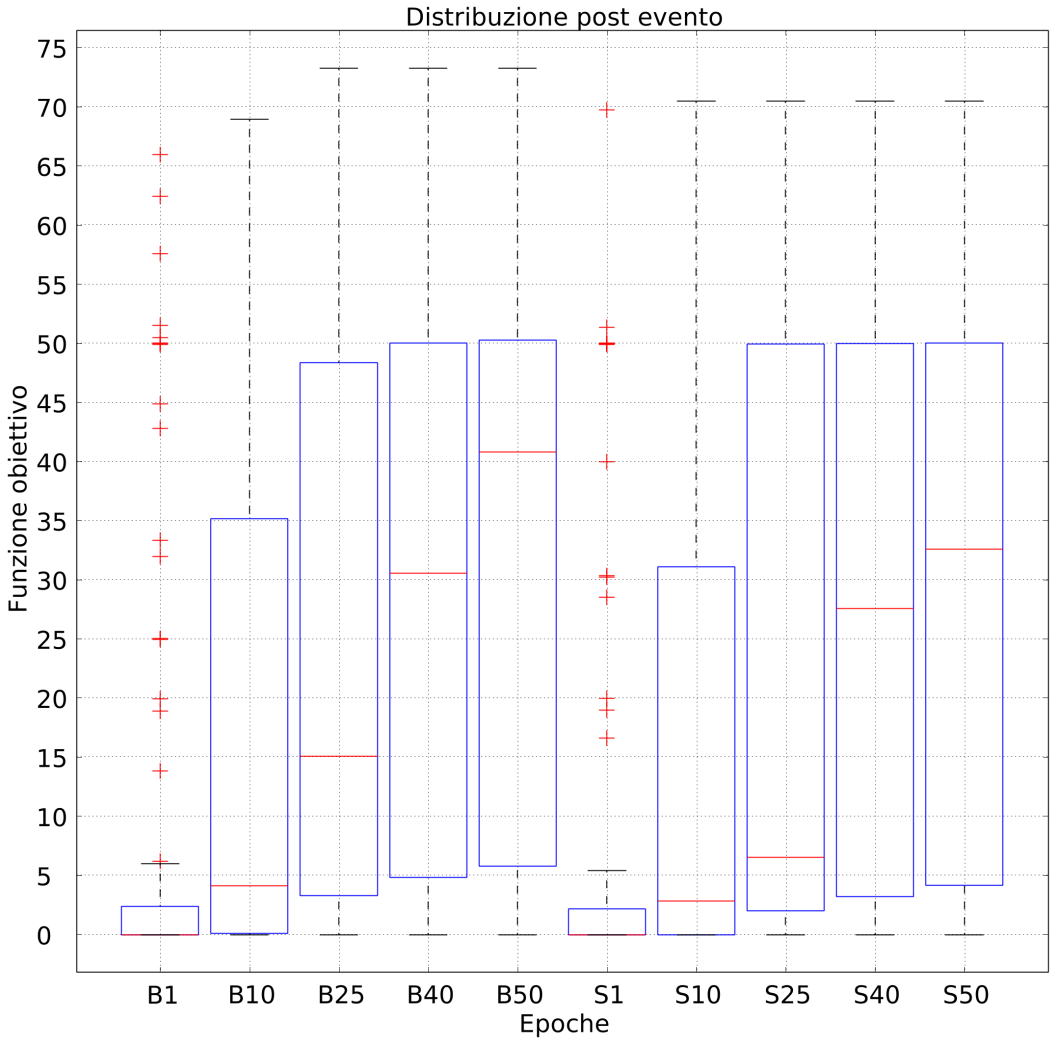


Figura 3.12: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente B come fase finale

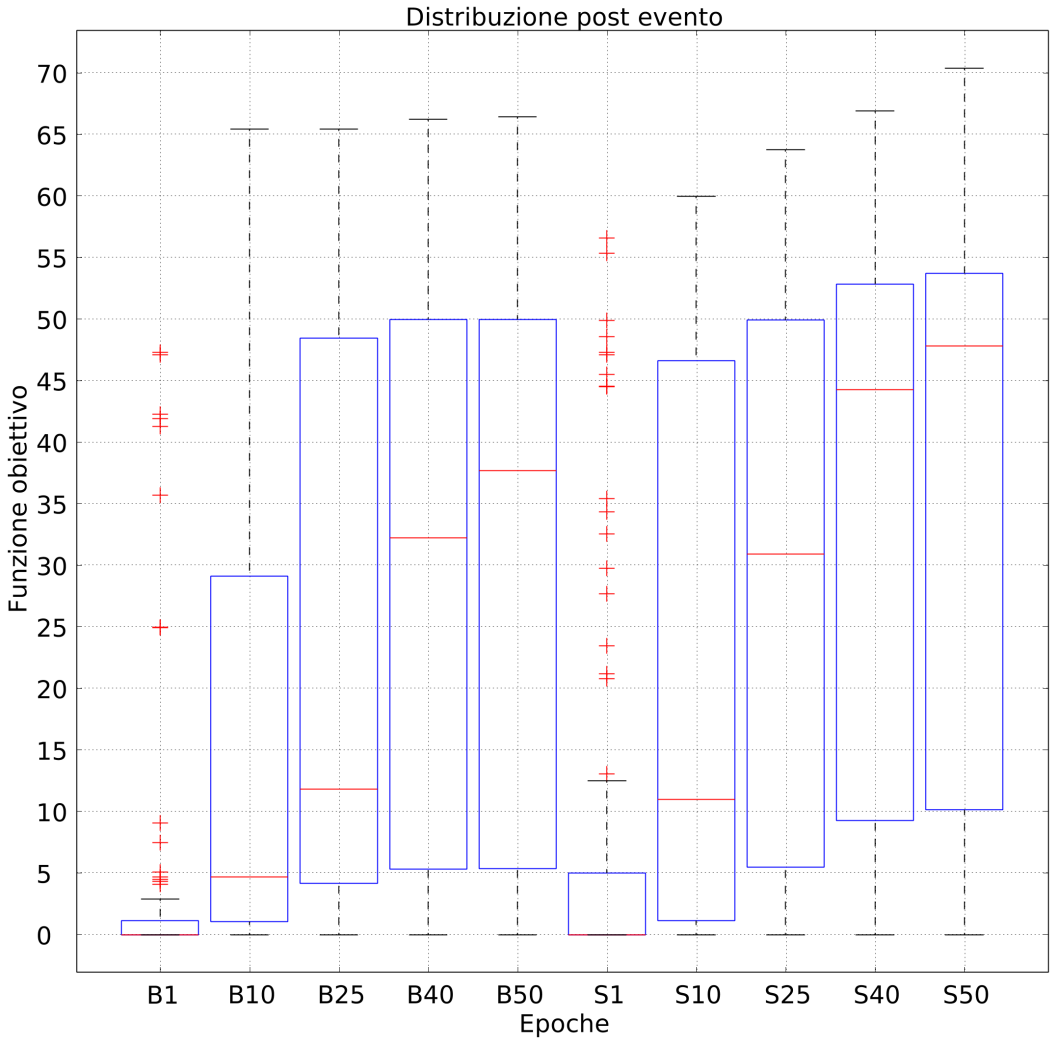
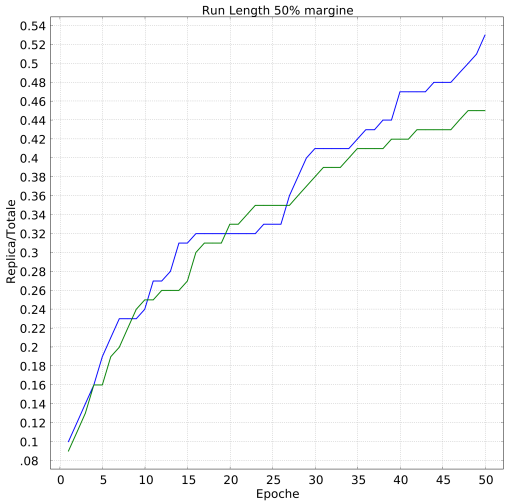


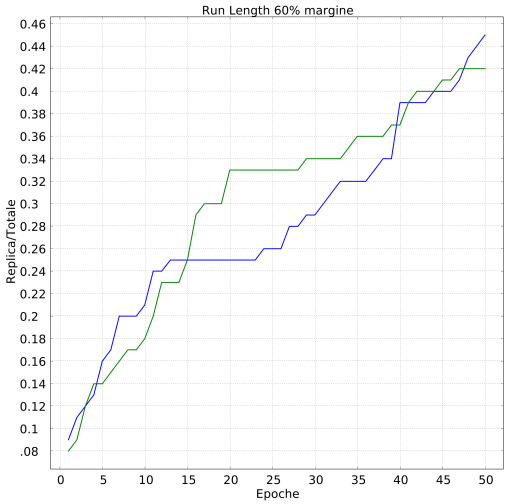
Figura 3.13: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente A come fase finale

In figura 3.14 si mette in luce il comportamento quantitativo ad ogni epoca degli esemplari adattati durante la simulazione, relativo all'ordine di simulazione AB. Il calcolo dei tre grafici mostra come l'andamento della serie relativa alla fase sperimentale sia perlopiù equivalente a quella di benchmark nel calcolo al 50% (figura 3.14a) e nel calcolo al 60%(3.14b). Nel calcolo al 70% (figura

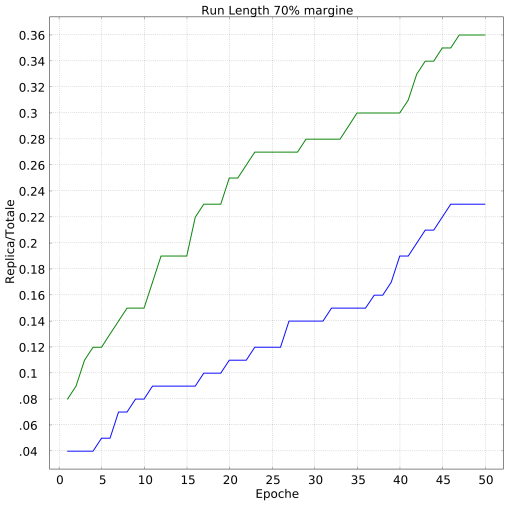
3.14c) si nota come l'andamento della funzione che delinea la rete preadattata domini completamente l'andamento della rete di benchmark.



(a) RLD tarate al 50%



(b) RLD tarate al 60%

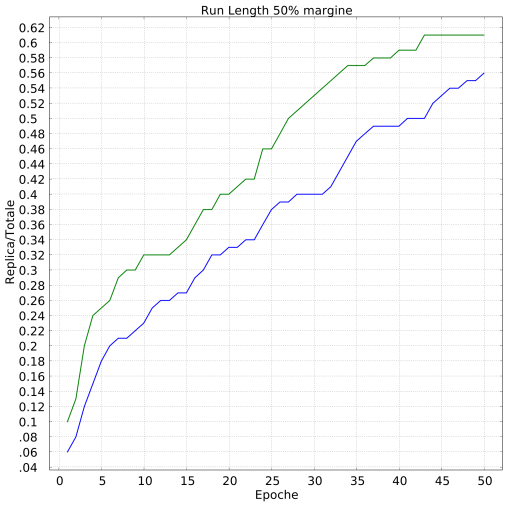


(c) RLD tarate al 70%

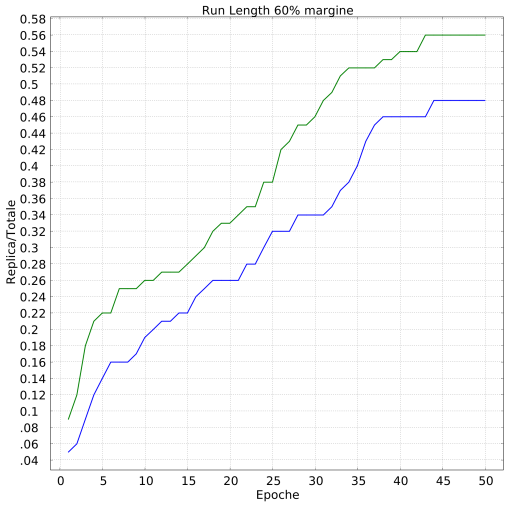
Figura 3.14: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale

In figura 3.15 si mette in luce il comportamento quantitativo degli esem-

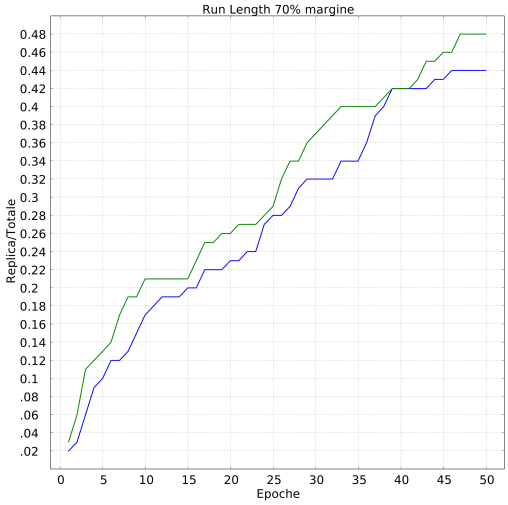
plari generati in ogni epoca dell'esperimento relativo all'ordine di simulazione BA. Il calcolo dei tre grafici mostra come l'andamento della serie relativa alla fase sperimentale sia sempre dominante rispetto all'andamento della fase di benchmark nei tre casi.



(a) RLD tarate al 50%



(b) RLD tarate al 60%



(c) RLD tarate al 70%

Figura 3.15: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale

Su tutte le simulazioni viene effettuato il test di indipendenza statistica di MannWhitney. Il valore di α per rifiutare l'ipotesi nulla si considera fissato a 0.05. Con *init* e *max* ci si riferisce rispettivamente alla distribuzione dei valori

ad epoca 0 e ai massimi valori raggiunti a fine simulazione. I termini affiancati da una b o una s rimandano rispettivamente a dei risultati di benchmark e sperimentazione.

I risultati mostrano come nella fase di benchmark ad arena singola si possa rifiutare l'ipotesi nulla per il processo di adattamento su entrambe le arene, mentre ciò non è possibile confrontando la distribuzione dei valori massimizzati raggiunti nelle due arene singolarmente.

serie 1	serie 2	*p-value*	H_0
init	max	$9.7367918362E-20$	Rifiutata
init	max	$1.9812655624E-21$	Rifiutata
max A	max B	0.9697893170	Non rifiutata

Effettuando i test sui valori della funzione obiettivo associata all'empowerment nella fase di tuning, si possono rifiutare entrambe le ipotesi nulle per i valori iniziali e finali

serie 1	serie 2	*p-value*	H_0
init	max	$1.6125186386E-18$	Rifiutata
init	max	$8.3019635154E-21$	Rifiutata

Nella fase di benchmark ad arena doppia si può rifiutare l'ipotesi nulla per entrambi i processi di adattamento, considerando come *init* la distribuzione dei valori alla prima epoca della prima arena, come *max* la distribuzione dei valori massimi al termine della simulazione nel secondo ambiente mentre ciò non è possibile confrontando la distribuzione dei valori massimizzati raggiunti nelle due arene singolarmente.

serie 1	serie 2	*p-value*	H_0
init A	max B	$2.8446747582E-19$	Rifiutata
init B	max A	$3.1773316496E-20$	Rifiutata
maxed AB	maxed BA	0.7591133651	Non rifiutata

Nella fase di simulazione sperimentale risulta confermato il rifiuto delle ipotesi nulle in entrambi i processi di adattamento, dove con *init A* (risp. *init B*) e *max A* (risp. *max B*) si intendono la prima epoca nel contesto della prima arena dell'esperimento e l'ultima epoca nella seconda fase. Si può rifiutare l'ipotesi nulla anche nel caso del confronto fra le distribuzioni di valori delle funzioni obiettivo massimizzate nelle due fasi finali dei due ordini dell'esperimento, indicati in tabella con *max AB* e *max BA*.

serie 1	serie 2	*p-value*	H_0
init A	max B	$1.2200477617E-22$	Rifiutata
init B	max A	$2.8659443599E-23$	Rifiutata
max AB	max BA	0.0364796051	Rifiutata

Effettuando il test statistico sulle distribuzioni di valori massimizzate nelle fasi di benchmark e sperimentazione, non si può rifiutare l'ipotesi nulla in nessuno degli ordini di adattamento.

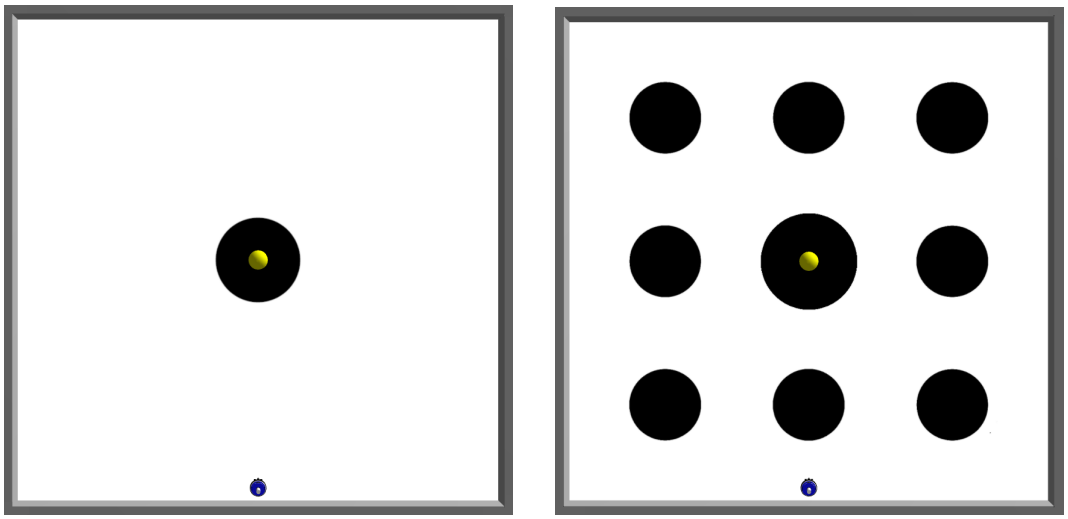
serie 1	serie 2	*p-value*	H_0
max AB(b)	max AB(s)	0.3770822615	Non rifiutata
max BA(b)	max BA(s)	0.0665079039	Non rifiutata

3.5 Ricerca spaziale assistita

L'esperimento di ricerca spaziale assistita da gradiente consiste nell'individuare e posizionarsi in una zona specifica dell'arena: ciò che contraddistingue tale zona dal resto dell'ambiente è l'emissione di un segnale luminoso di intensità inversamente proporzionale alla distanza dalla sorgente.

Setup ambiente Gli ambienti sono costituiti da arene di dimensione 5x5, all'interno delle quali il robot è vincolato da un perimetro che risiede sul margine di tale area. Il pavimento delle arene è costituito da zone bianche e zone

nere. Una sola di queste zone emette un segnale luminoso che irradia un'area circolare intorno a sè con diametro pari a due terzi del lato dell'arena. Il robot si suppone nascere in una zona non inizialmente colpita dalla luce, nè corrispondente alla zona obiettivo.



(a) Ambiente A (b) Ambiente B

Figura 3.16: Coppia di ambienti nei quali è stata effettuata l'esperimento di ricerca assistita

Setup robot Il robot è equipaggiato con quattro sensori di luminosità, posizionati in corrispondenza dei punti cardinali fondamentali, e quattro sensori di terreno, disposti sui punti cardinali collaterali. I sensori di luminosità considerano luce ogni percezione superiore ad un *threshold* pari ad una percentuale della propria risoluzione massima, restituendo perciò un valore dagli intervalli [0, *threshold*) i cui valori sono tutti considerati come 0, e l'intervallo [*threshold*, 1], dove 0 indica l'assenza di percezione e 1 la saturazione del sensore. I sensori di terreno restituiscono 1 nel caso in cui rilevino una patch bianca, 0 una nera.

Funzione obiettivo La funzione che il processo euristico di adattamento intende massimizzare consiste in

$$f_{epoch} = \sum_{1}^{t} (H_l * W_l + N_g * W_g)/(W_l + W_g)$$

dove:

- t rappresenta il numero totale di passi per ogni epoca della simulazione;
- H_l rappresenta il massimo valore percepito dai sensori di luce;
- W_l rappresenta il peso associato al valore dei sensori di luminosità
- N_g rappresenta il numero di sensori di terreno attivi al momento;
- W_g rappresenta il peso associato al valore dei sensori di terreno.

L'introduzione di fattori pesati è stata necessaria per bilanciare la differente cardinalità dei valori percepiti.

Lo scopo di tale funzione è quello di premiare, nell'intera epoca del processo di adattamento, la vicinanza alla sorgente luminosa e nel caso in cui il robot riesca a posizionarsi sulla zona di effettivo interesse un premio aggiuntivo che consolidi la qualità del comportamento.

Setup simulazione Il simulatore effettua 100 repliche dell' esperimento, ognuna delle quali consiste in 50 epoche di adattamento divise in 2500 *tick* dell'ambiente, che nella rappresentazione interna del tempo logico del simulatore corrispondono a 250 secondi di tempo effettivo. Un'intera replica dell'esperimento si considera quindi equivalere a 3 ore, 28 minuti e 20 secondi di tempo effettivo.

Nel caso delle simulazioni ad ambiente multiplo il numero di repliche si considera raddoppiato, in quanto l'adattamento viene ripetuto per intero sulla seconda arena una volta terminato il primo ciclo.

La percentuale di *threshold* luminoso oltre la quale il robot percepisce la luce è fissata al 10%.
W_l, dalla formula della funzione obiettivo, è imposto a 8.
W_g, dalla formula della funzione obiettivo, è imposto a 0.4.

I parametri del processo di inizializzazione della RBN di controllo sono fissati secondo i parametri standard delle RBN di tipo criticale[6]:

- 100 nodi complessivi;
- 3 connessioni entranti per ogni nodo;
- un *bias* di generazione delle funzioni booleane per ogni nodo pari a 0,79.

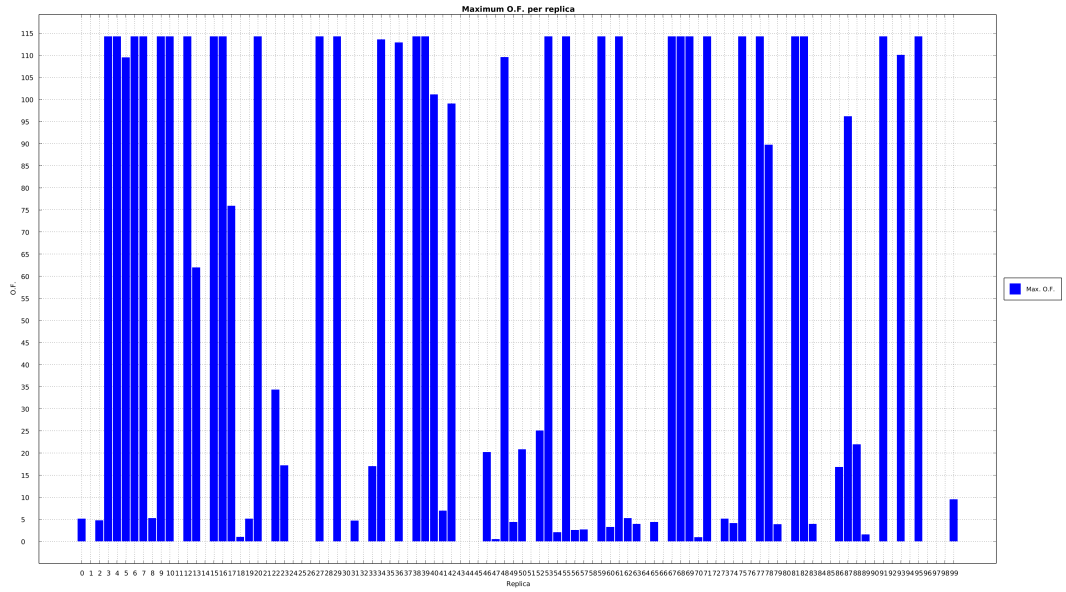
Inoltre, gli altri parametri di generazione sono stati fissati come segue:

- 8 nodi di input;
- 2 nodi di output;
- un *bias* di generazione delle funzioni booleane relative all'attivazione dell'output pari a 0.5 [4];
- 3 possibili *rewiring* dei nodi di input nella fase di adattamento all'ambiente.

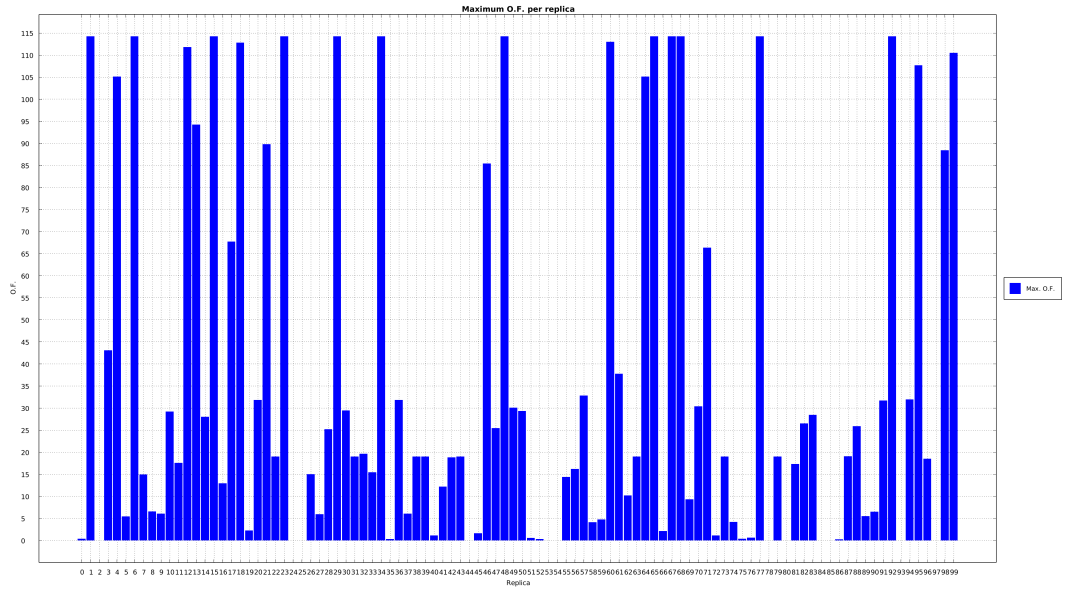
3.5.1 Benchmark singola arena

Il benchmark dell'adattamento tramite sola funzione obiettivo taglia chiaramente in due il panorama di possibilità in cui possono ricadere i risultati raggiunti dalla RBN. Essendo un task esplorativo incrementale, una delle possibilità è che il robot non riesca a raggiungere lo stimolo luminoso che innescherebbe il processo di ricerca, ottenendo in questo caso un valore di funzione

obiettivo infimo. Al contrario individuare la zona luminosa porta quasi sempre al massimo obiettivo possibile. Questo resta vero anche nell'adattamento effettuato nella seconda arena, anche se l'introduzione di rumore che ostacoli il diretto svolgimento del task diminuisce in maniera proporzionale gli esemplari che equiparano le prestazioni nel primo ambiente. Questi tre cluster concettuali sono visualizzabili negli istogrammi in figura 3.17, che mostrano i valori massimi delle funzioni obiettivo raggiunti nelle diverse repliche dello stesso esperimento.



(a) Valori raggiunti nell'arena A



(b) Valori raggiunti nell'arena B

Figura 3.17: Istogrammi dei valori massimi della funzione obiettivo in ogni replica del benchmark ad arena singola

I boxplot in figura 3.18 rispecchiano questo comportamento: in particolare

il boxplot relativo all'ambiente B quantifica la difficoltà incontrata nella seconda arena nel raggiungere l'obiettivo effettivo, mostrando come prominente la distribuzione sui bassi valori del rumore non obiettivo. Ciò appare evidente, oltre che dagli istogrammi già presentati, dalla presenza dei valori massimali raggiunti come facenti parte del quarto quartile del diagramma a scatola in figura.

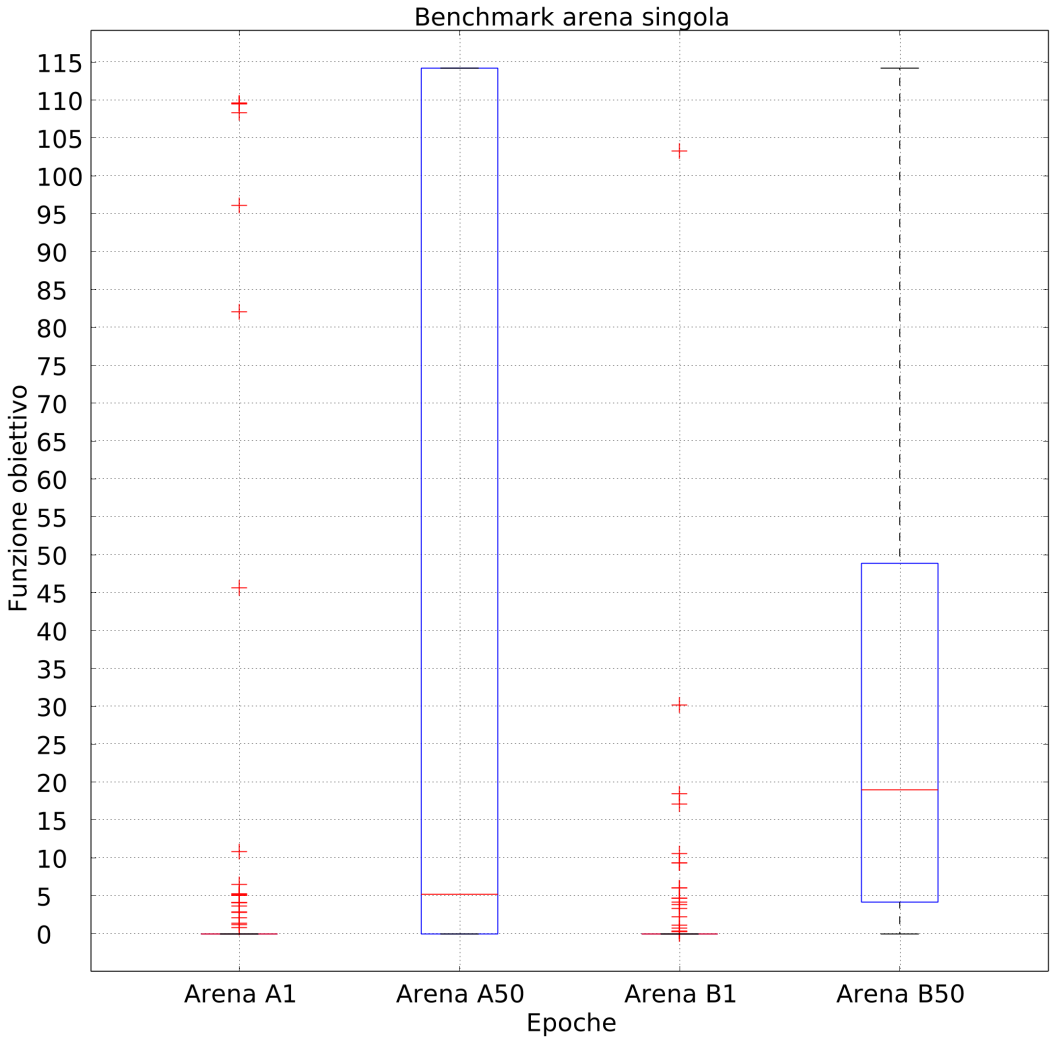
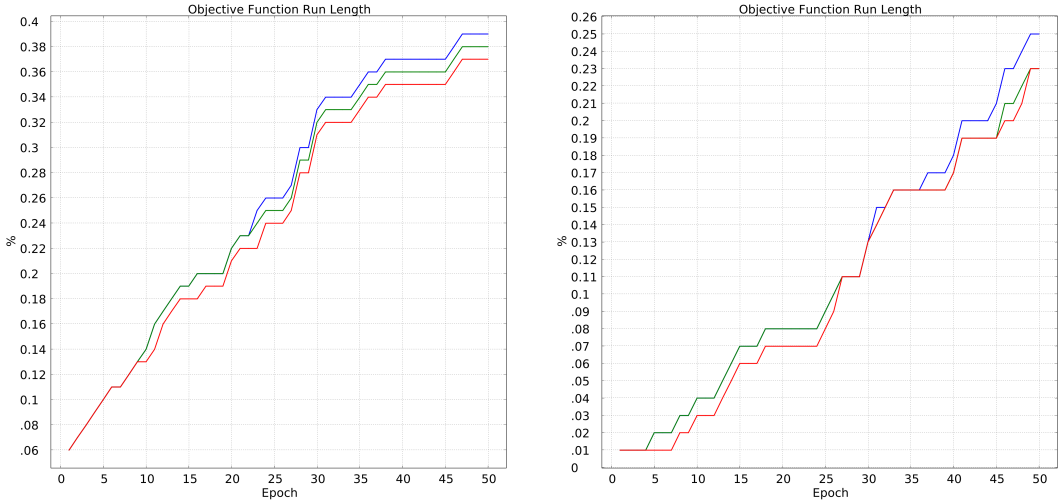


Figura 3.18: Nella metà sinistra (risp. destra), la distribuzione dei valori di funzione obiettivo di benchmark per la sola arena A (risp. B)

Attraverso il plotting delle *run length distribution* viene riconfermato come tutte le distribuzioni appartenenti ad una RBN che ha individuato la luce arrivino tutte a soglie molto alte di valori di funzione obiettivo, nello specifico il 37% nella prima arena mentre solo il 23% nella seconda, a riprova della maggiorata difficoltà nel navigare l'arena dovuta all'introduzione del rumore.



(a) RLD su ambiente A

(b) RLD su ambiente B

Figura 3.19: *Run length distribution* degli ambienti di benchmark ad arena singola: le tracce blu, verdi e rosse rappresentano rispettivamente la soglia di calcolo posta al 50, 60 e 70 per cento del valore massimo raggiunto

3.5.2 Tuning singola arena

Per l'ambiente A il valore di *cut-off* si è consolidato a 2, valore con il quale sarà inizializzato l'esperimento che ha questo ambiente come iniziale. Il valore di *cut-off* dell'ambiente B è stato rilevato come 3, ed è il valore dopo il quale si tornerà all'adattamento tramite funzione obiettivo nella simulazione sperimentale.

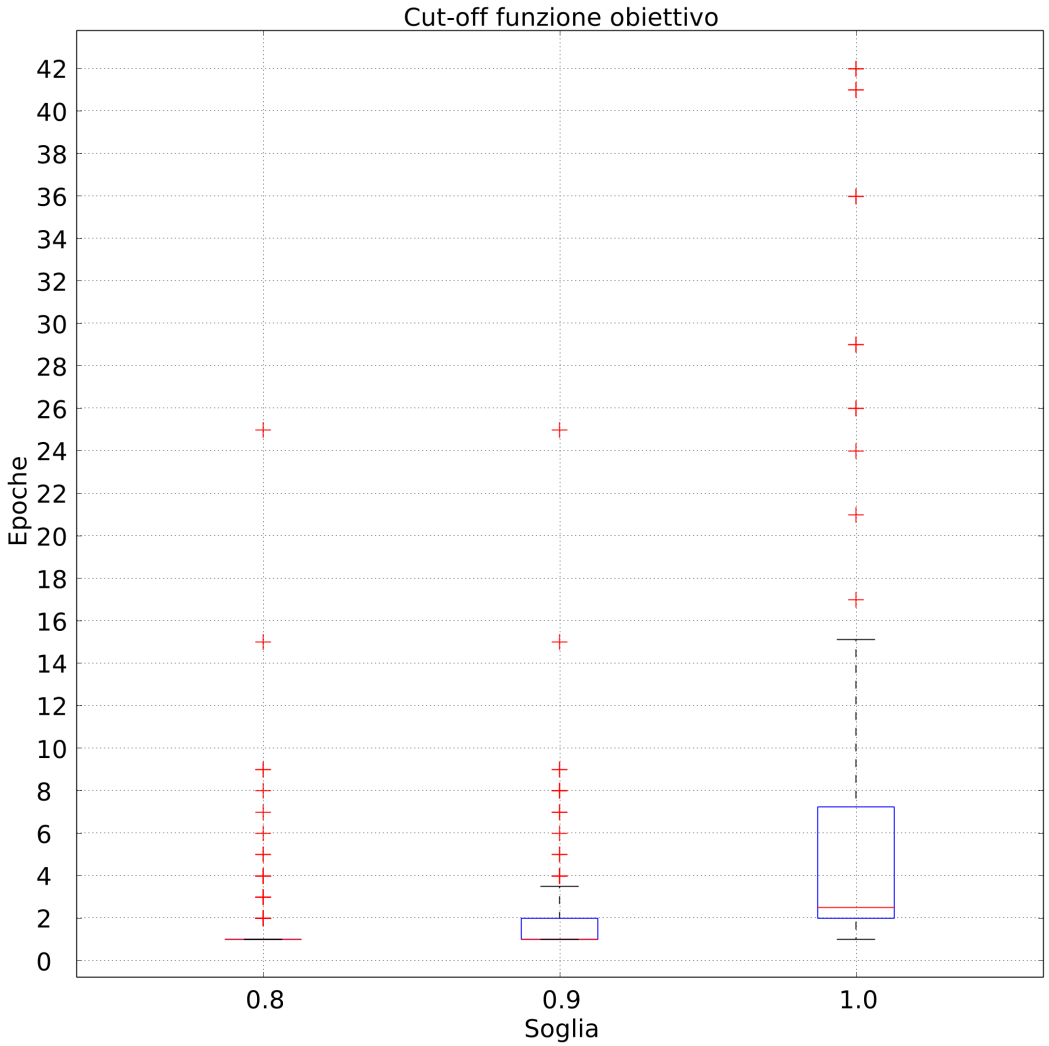


Figura 3.20: Soglie di cutoff su ambiente A

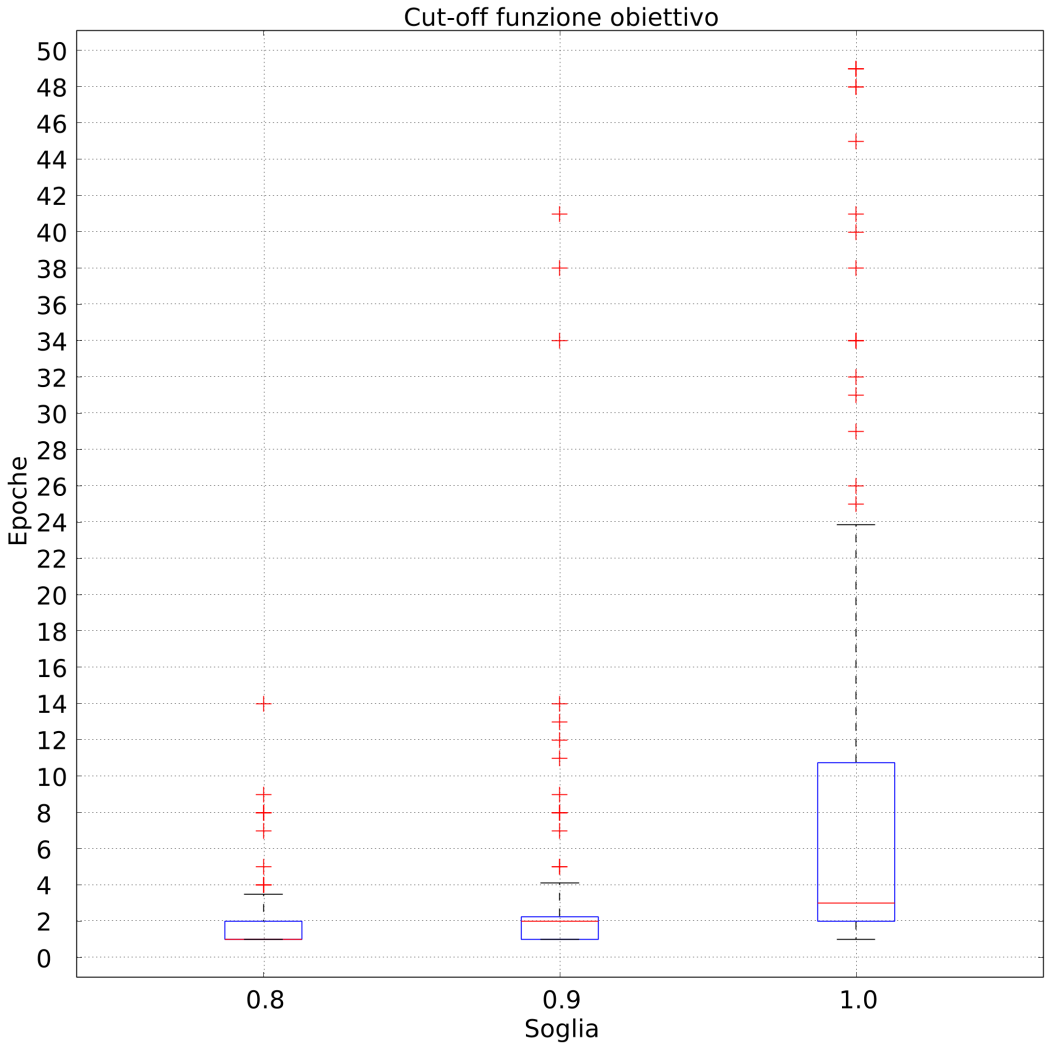


Figura 3.21: Soglie di cutoff su ambiente B

I valori della funzione obiettivo associati all'adattamento tramite massimizzazione dell'empowerment hanno registrato valori tendenzialmente bassi rispetto alla controparte con utilizzo della funzione obiettivo come motore dell'adattamento, ma comunque in linea con l'andamento generale all'interno della propria arena di competenza. L'arena B ha registrato bassi livelli di funzione obiettivo anche quando lo scopo dell'adattamento era massimizzarne il valore.

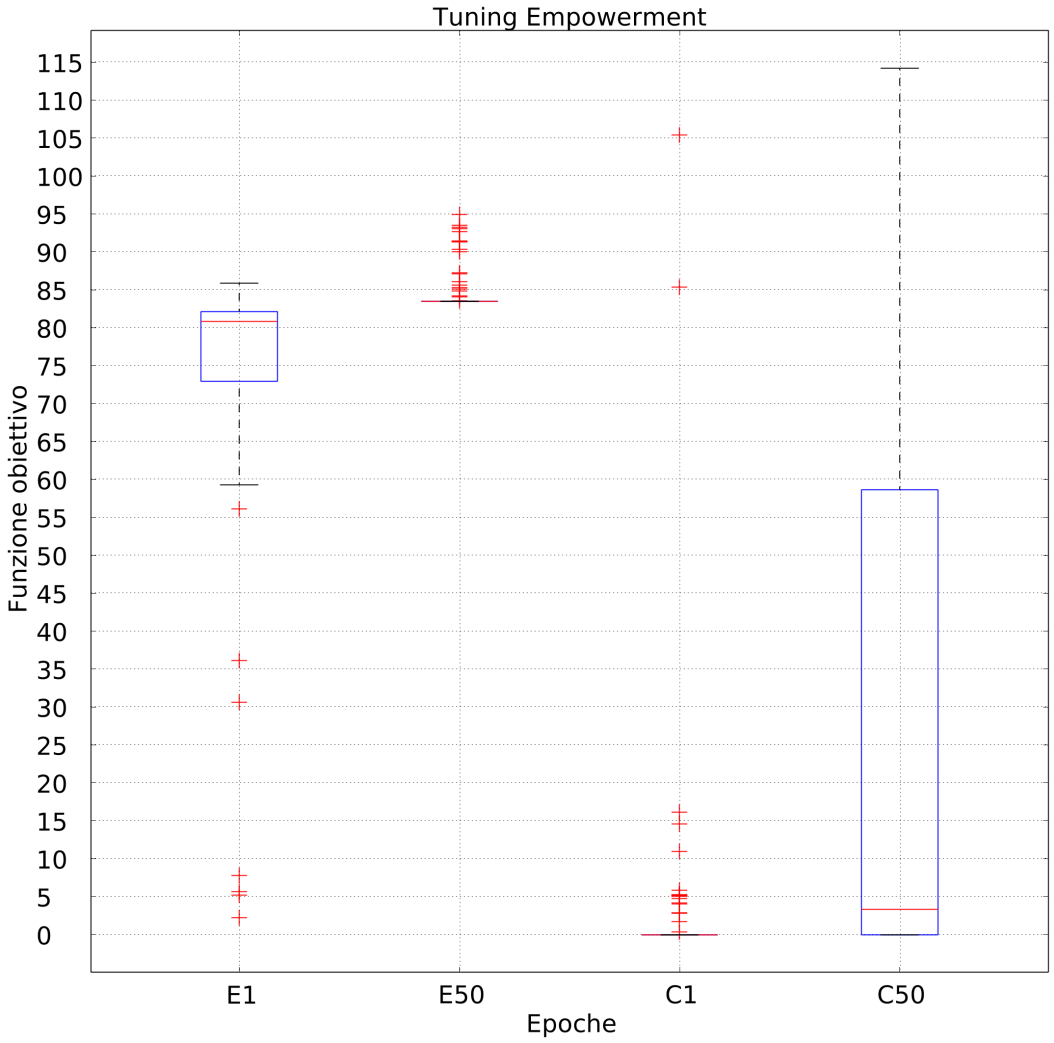


Figura 3.22: Distribuzioni ad epoca 1 e 50 delle funzioni di empowerment (sinistra) e obiettivo (destra) su ambiente A

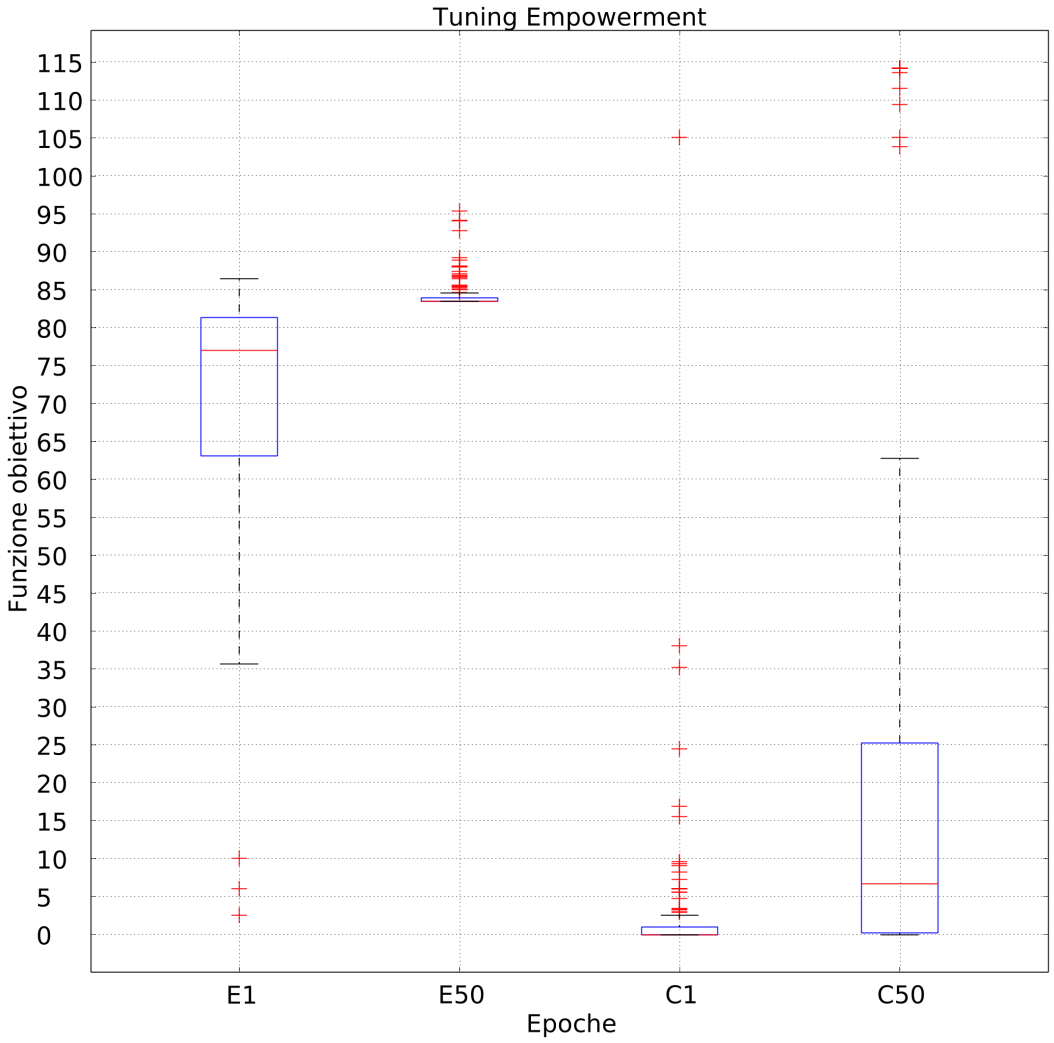


Figura 3.23: Distribuzioni ad epoca 1 e 50 delle funzioni di empowerment (sinistra) e obiettivo (destra) su ambiente B

3.5.3 Benchmark sperimentale

Il trasferimento ddel migliore esemplare dall'arena A all'arena B è riuscito a mantenere una distribuzione di valori della funzione obiettivo molto alto, quadruplicando la mediana dei valori. Lo stesso non vale per il trasferimento inverso, in quanto il trasferimento del miglior esemplare dall'arena B all'arena

A ha abbattuto la qualità del risultato, pur mantenendo la mediana della distribuzione.

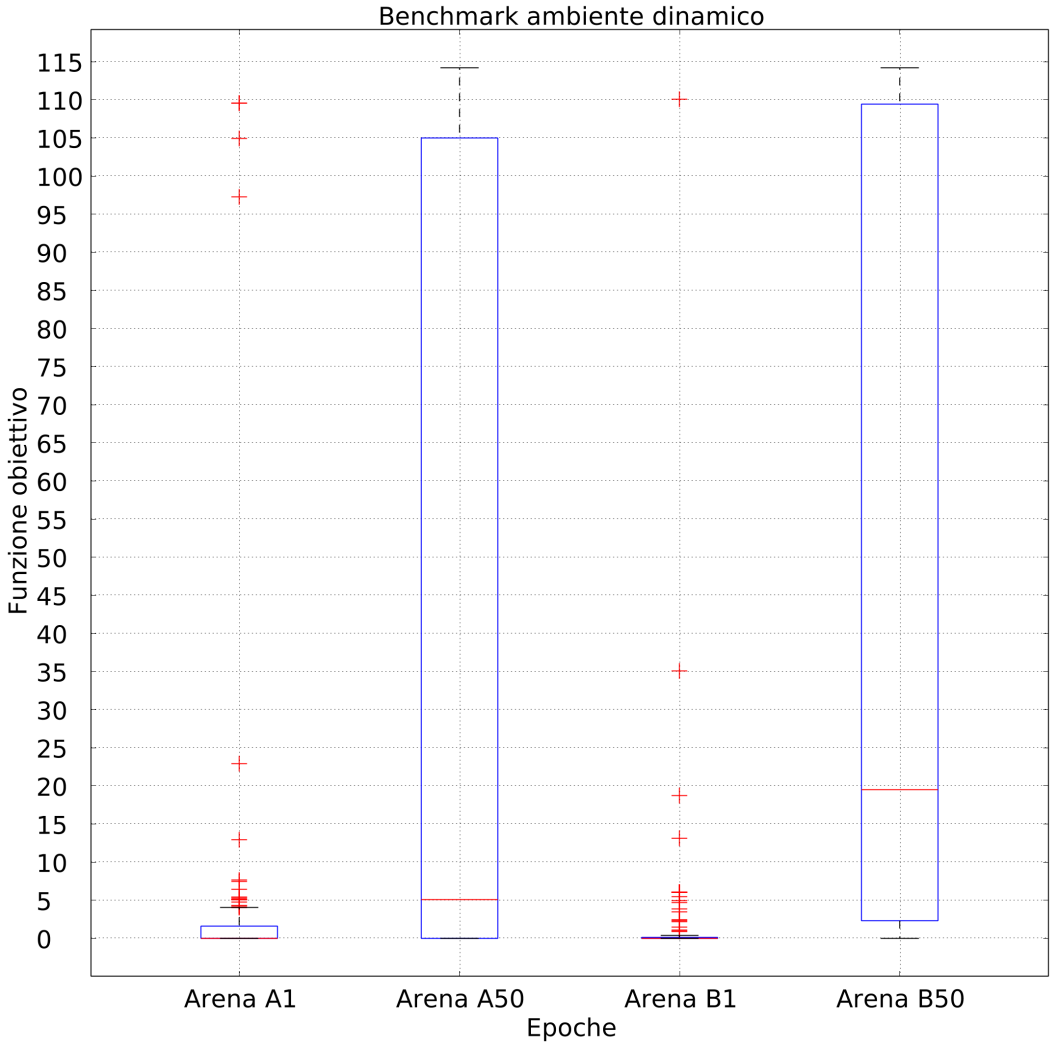


Figura 3.24: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena A, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena A nell'arena B

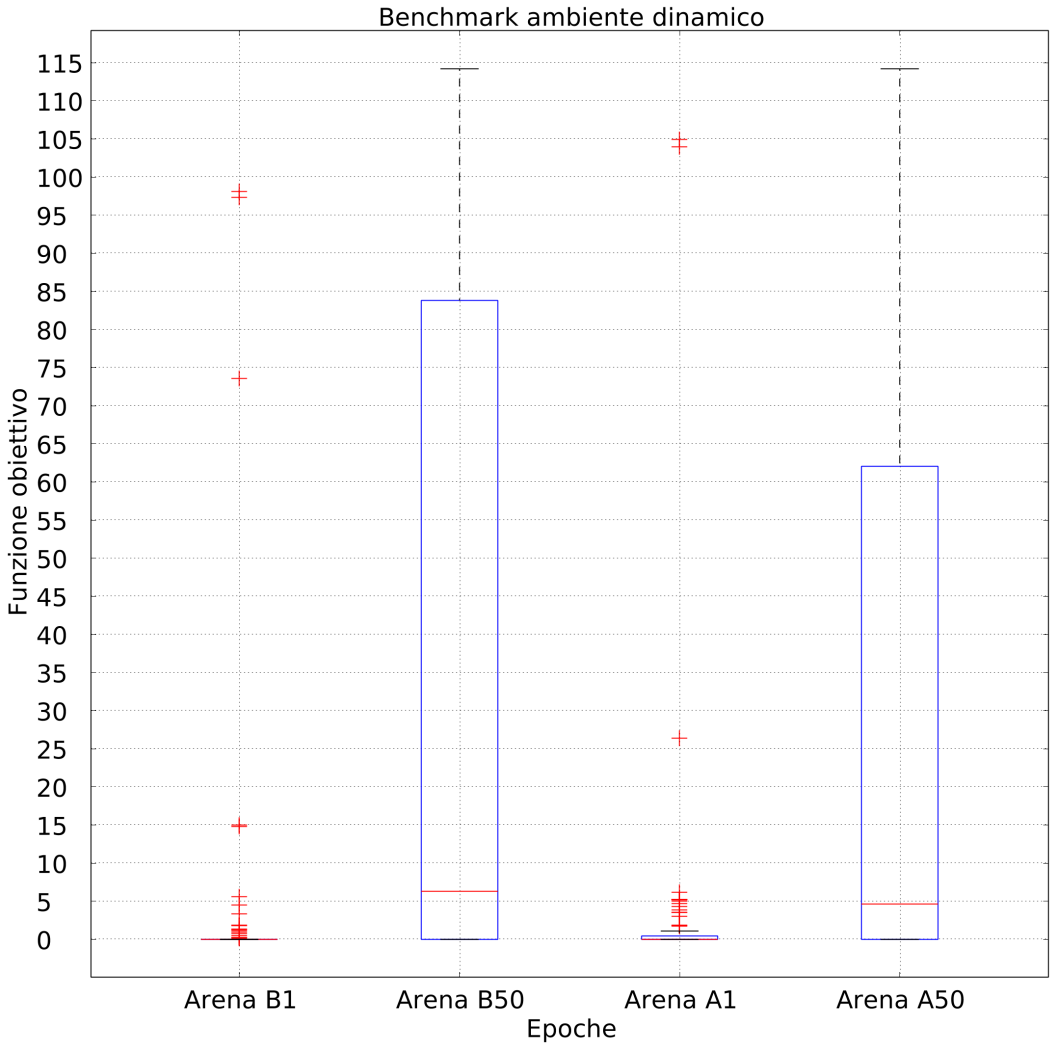


Figura 3.25: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena B, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena B nell'arena A

3.5.4 Sperimentazione

Il passaggio da ambiente A con preadattamento ad ambiente B ha portato ad un incremento relativo sia in termini di valori della distribuzione risultante, che in mediana della distribuzione. Questo tuttavia è successo in seguito ad

un notevole calo delle prestazioni nella prima arena, che ha registrato valori particolarmente bassi.

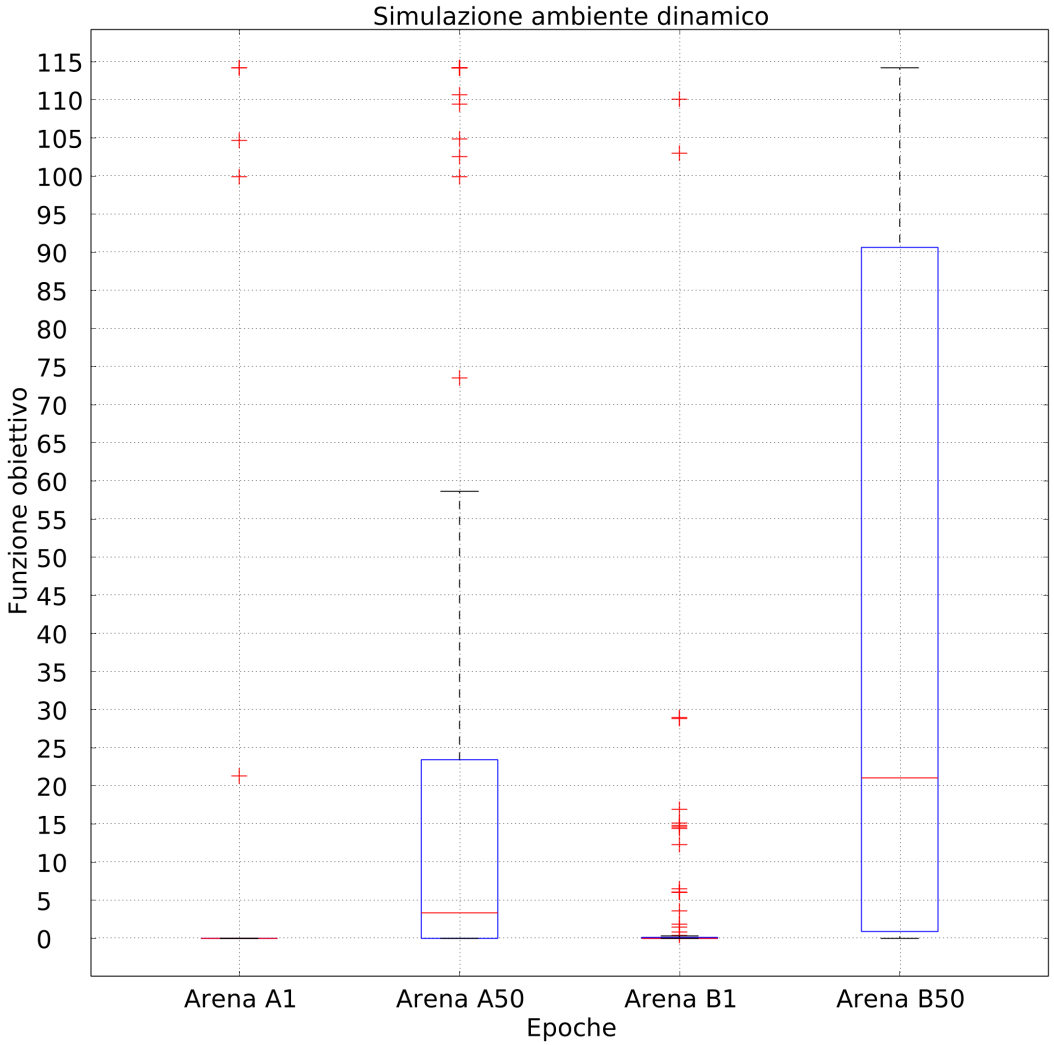


Figura 3.26: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena A, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena B

Effettuando il preadattamento nell'arena B, nonostante un adattamento decisamente positivo per gli standard dell'ambiente, al momento del trasferi-

mento si è riscontrato un decremento della mediana di distribuzione dei valori per quanto il massimale di funzione obiettivo sia stato ripristinato.

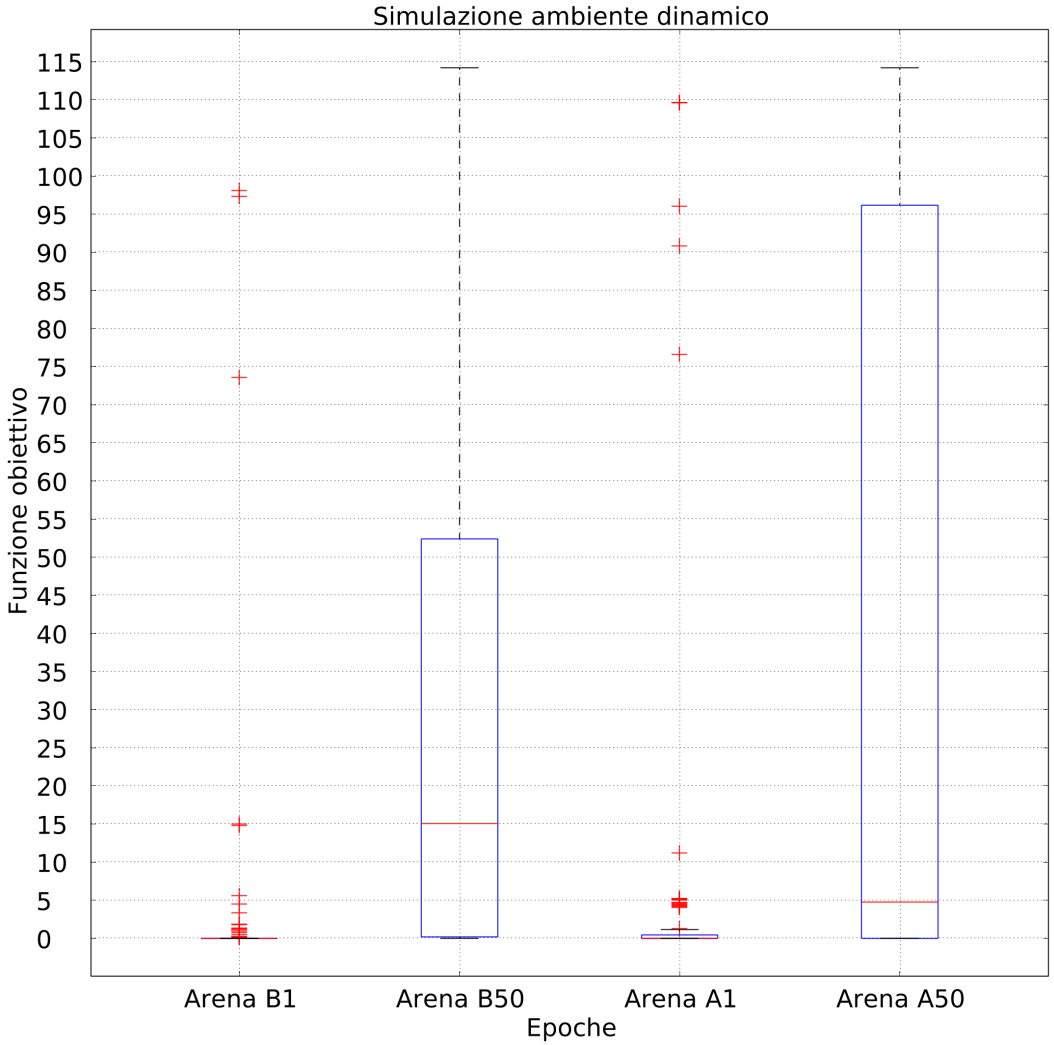


Figura 3.27: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena B, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena A

3.5.5 Risultati

Il confronto fra le distribuzioni dei valori ottenute in benchmark e fase sperimentale nella simulazione AB (pre-adattamento in A con trasferimento in B) mostra come nella fase finale dell'esperimento i risultati ottenuti in benchmark non riescano a venire eguagliati da quelli ottenuti in fase sperimentale, nonostante il valore delle mediane delle distribuzioni assuma dei valori assimilabili a quelli della fase di benchmark.

Come già successo nell'esperimento di *Navigazione fra ostacoli* questo risultato viene contraddetto dal confronto della simulazione in cui l'ordine delle arene viene invertito. In questo caso (figura 3.29) esiste una netta superiorità dei valori della distribuzione della fase sperimentale, i quali subiscono un incremento molto più alto, molto più velocemente, della fase di benchmark. Le mediane rimangono stabili per l'intero processo di adattamento.

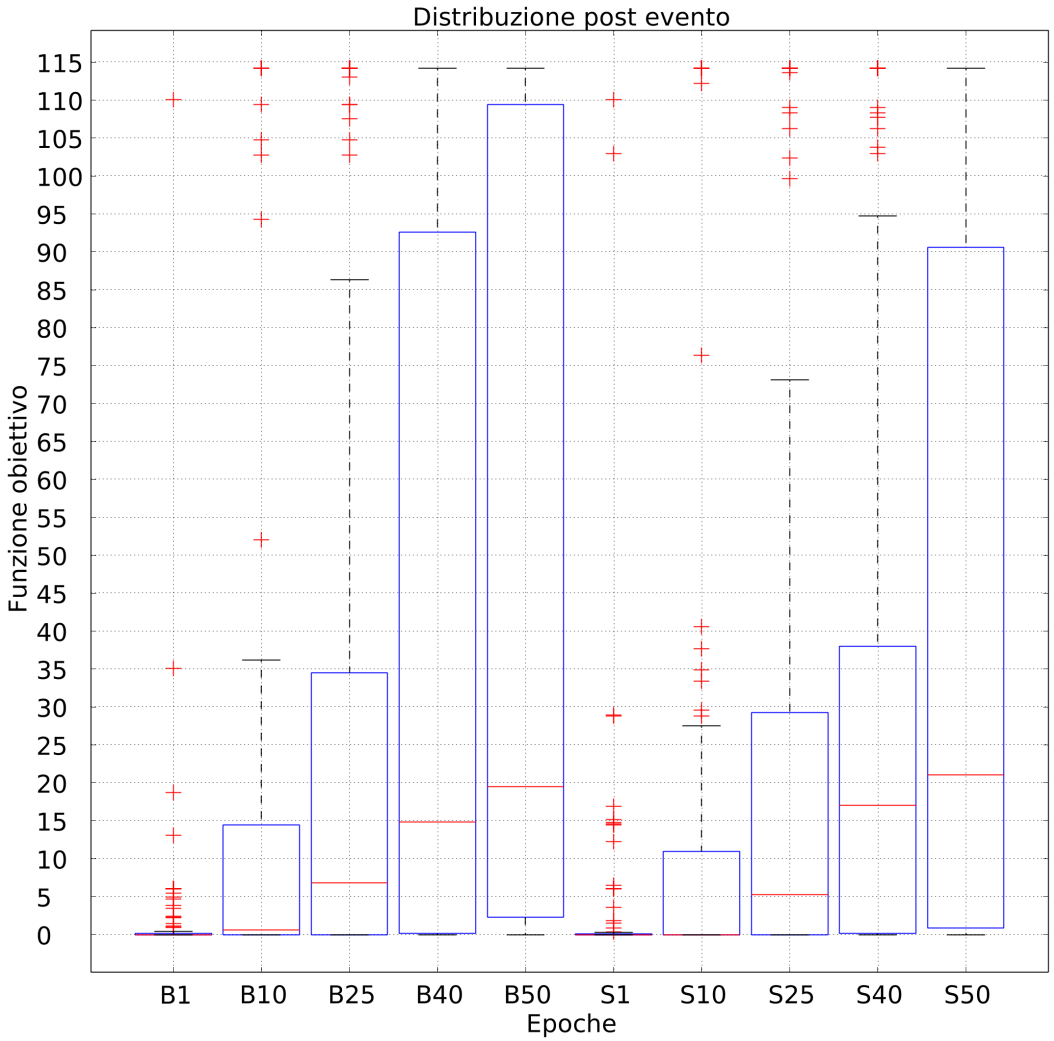


Figura 3.28: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente B come fase finale

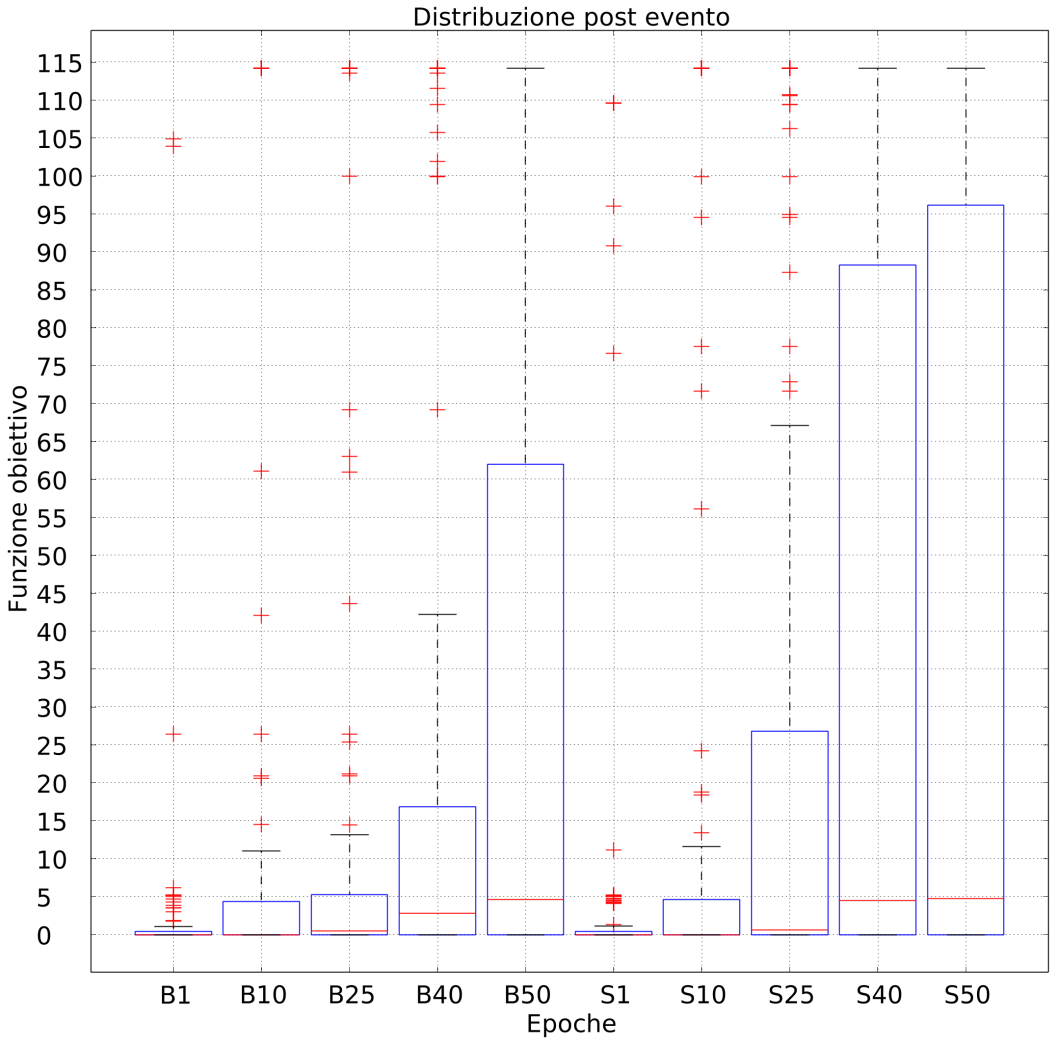
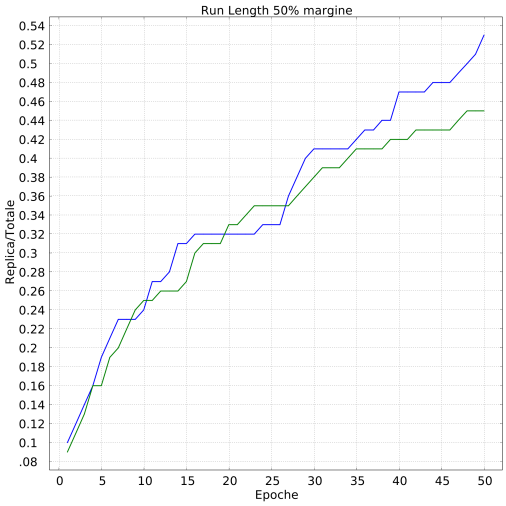
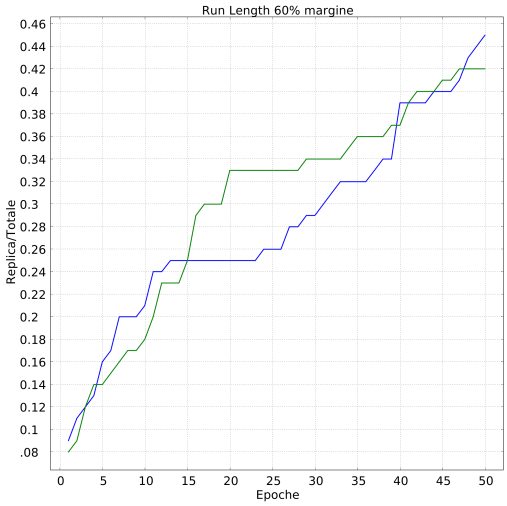


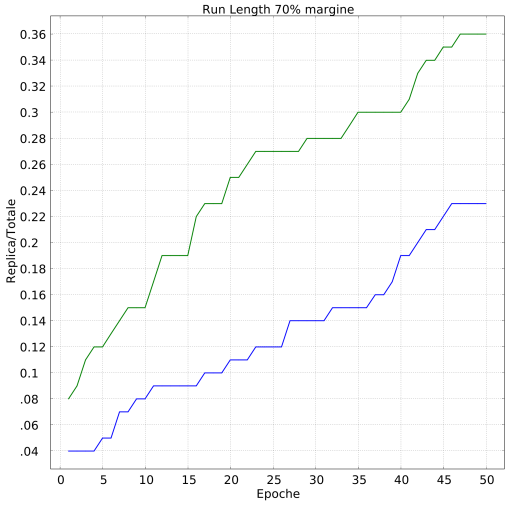
Figura 3.29: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente A come fase finale



(a) RLD tarate al 50%

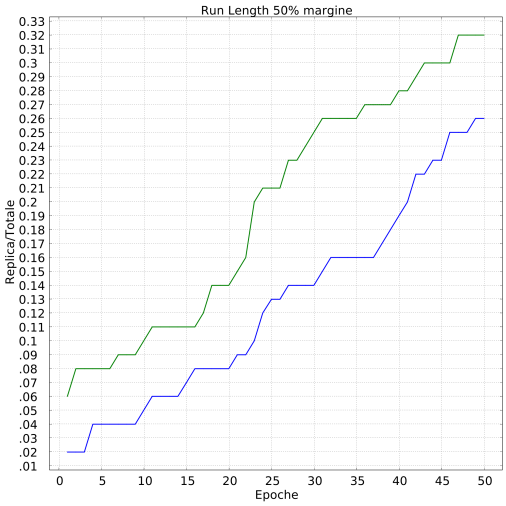


(b) RLD tarate al 60%

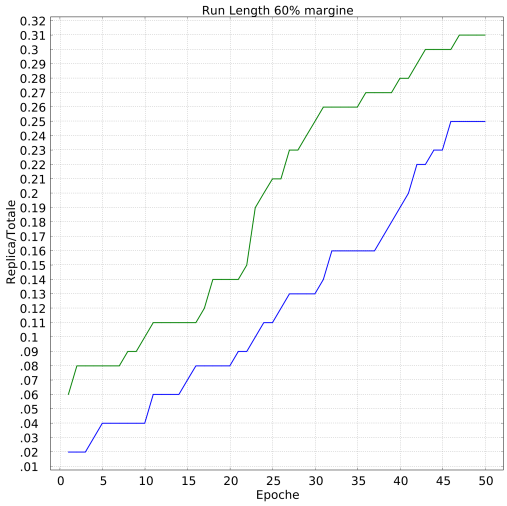


(c) RLD tarate al 70%

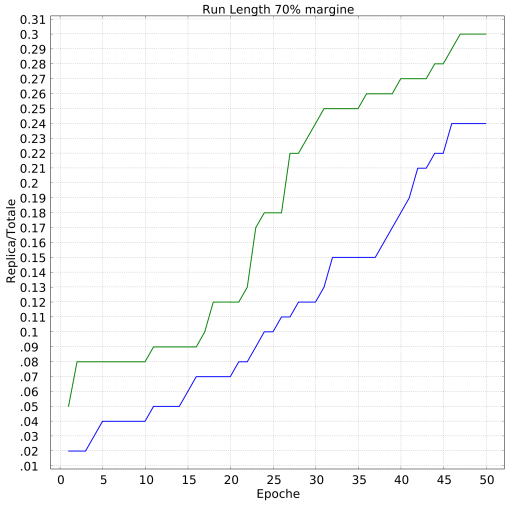
Figura 3.30: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale



(a) RLD tarate al 50%



(b) RLD tarate al 60%



(c) RLD tarate al 70%

Figura 3.31: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale

Su tutte le simulazioni viene effettuato il test di indipendenza statistica di MannWhitney. Il valore di α per rifiutare l'ipotesi nulla si considera fissato a 0.05. Con *init* e *max* ci si riferisce rispettivamente alla distribuzione dei valori

ad epoca 0 e ai massimi valori raggiunti a fine simulazione. I termini affiancati da una b o una s rimandano rispettivamente a dei risultati di benchmark e sperimentazione.

I risultati mostrano come nella fase di benchmark ad arena singola si possa rifiutare l'ipotesi nulla per il processo di adattamento su entrambe le arene, mentre ciò non è possibile confrontando la distribuzione dei valori massimizzati raggiunti nelle due arene singolarmente.

serie 1	serie 2	*p-value*	H_0
init	max	$5.496210719023982E-11$	Rifiutata
init	max	$2.6870854997168352E-21$	Rifiutata
max A	max B	0.48926453867926445	Non rifiutata

Effettuando i test sui valori della funzione obiettivo associata all'empowerment nella fase di tuning, si possono rifiutare entrambe le ipotesi nulle per i valori iniziali e finali

serie 1	serie 2	*p-value*	H_0
init	max	$5.189826646279343E-11$	Rifiutata
init	max	$7.704835891277118E-12$	Rifiutata

Nella fase di benchmark ad arena doppia si può rifiutare l'ipotesi nulla per entrambi i processi di adattamento, considerando come *init* la distribuzione dei valori alla prima epoca della prima arena, come *max* la distribuzione dei valori massimi al termine della simulazione nel secondo ambiente. Confrontando la distribuzione dei valori massimizzati raggiunti nelle due arene singolarmente, è anche possibile rifiutare l'ipotesi nulla anche in questo caso.

serie 1	serie 2	*p-value*	H_0
init A	max B	$1.3854643315584787E-16$	Rifiutata
init B	max A	$1.0321996316469156E-12$	Rifiutata
maxed AB	maxed BA	0.004302329155459723	Rifiutata

Nella fase di simulazione sperimentale risulta confermato il rifiuto delle ipotesi nulle in entrambi i processi di adattamento, dove con *init A* (risp. *init B*) e *max A* (risp. *max B*) si intendono la prima epoca nel contesto della prima arena dell'esperimento e l'ultima epoca nella seconda fase. Non è possibile rifiutare l'ipotesi nulla nel caso del confronto fra le distribuzioni di valori delle funzioni obiettivo massimizzate nelle due fasi finali dei due ordini dell'esperimento, indicati in tabella con *max AB* e *max BA*.

serie 1	serie 2	*p-value*	H_0
init A	max B	$1.0772545374866224E-19$	Rifiutata
init B	max A	$2.0602541272851926E-13$	Rifiutata
max AB	max BA	0.07527467128434866	Non rifiutata

Effettuando il test statistico sulle distribuzioni di valori massimizzate nelle fasi di benchmark e sperimentazione, non si può rifiutare l'ipotesi nulla in nessuno degli ordini di adattamento.

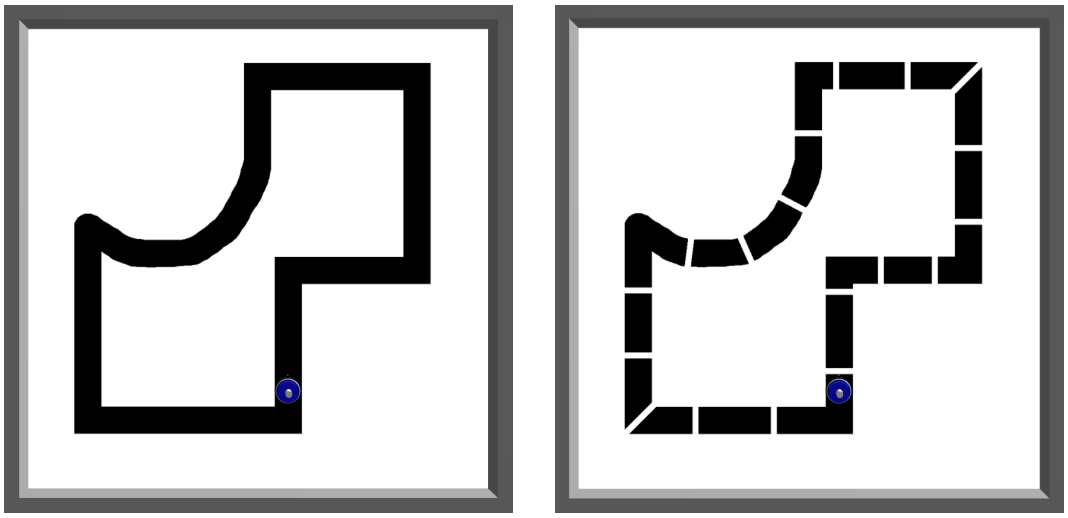
serie 1	serie 2	*p-value*	H_0
max AB(b)	max AB(s)	0.6886293933362203	Non rifiutata
max BA(b)	max BA(s)	0.6583058563114751	Non rifiutata

3.6 Navigazione su tracciato

L'esperimento di *Navigazione su tracciato* consiste nel mantenere la traiettoria del robot al di sopra di un tracciato impresso sul pavimento dell'arena.

Setup ambiente Gli ambienti sono costituiti da arene di dimensione 3x3, all'interno delle quali il robot è vincolato da un perimentro che risiede sul margine di tale area. Sul pavimento è presente una traccia di colore diverso dal resto dell'area. Il robot viene fatto nascere in corrispondenza di una sezione

rettilinea del circuito. I robot sono inizializzati puntando verso il lato disposto a nord.



(a) Ambiente A (b) Ambiente B

Figura 3.32: Coppia di ambienti nei quali è stata effettuata la simulazione

Setup robot Il robot è equipaggiato con quattro sensori di terreno posizionati in corrispondenza dei quattro punti cardinali collaterali. Ognuno dei sensori restituisce 1 quando si trova su una zona bianca, 0 quando viene rilevata una zona nera.

Funzione obiettivo La funzione che il processo euristico di adattamento intende massimizzare consiste in

$$f_{epoch} = \sum_{t} max^{*}(\overrightarrow{p(0), p(t)}) \cdot g(t) \cdot v(t) \tag{3.1}$$

dove:

- t rappresenta il numero totale di passi per ogni epoca della simulazione;
- $max^{*}(\overrightarrow{p(0), p(t)})$ rappresenta la distanza percorsa dal robot, calcolata a partire dalla posizione registrata ad inizio epoca.

- $g(t)$ rappresenta la presenza del robot su di una patch nera, e assume i valori 0 o 1 a seconda della risposta dei sensori.

- $v(t)$ rappresenta l'attivazione degli attuatori del robot, e assume i valori 0 o 1 a seconda che almeno una delle ruote sia attiva.

Lo scopo di tale funzione è quello di premiare, nell'intera epoca del processo di adattamento, un comportamento che si sposti molto lungo il percorso durante l'epoca. Restare semplicemente fermo azzera la funzione, così come non trovarsi entro i limiti del tracciato.

Setup simulazione Il simulatore effettua 100 repliche dell' esperimento, ognuna delle quali consiste in 50 epoche di adattamento divise in 2500 *tick* dell'ambiente, che nella rappresentazione interna del tempo logico del simulatore corrispondono a 250 secondi di tempo effettivo. Un'intera replica dell'esperimento si considera quindi equivalere a 3 ore, 28 minuti e 20 secondi di tempo effettivo.

Nel caso delle simulazioni ad ambiente multiplo il numero di repliche si considera raddoppiato, in quanto l'adattamento viene ripetuto per intero sulla seconda arena una volta terminato il primo ciclo.

I parametri del processo di inizializzazione della RBN di controllo sono fissati secondo i parametri standard delle RBN di tipo criticale[6]:

- 100 nodi complessivi;

- 3 connessioni entranti per ogni nodo;

- un *bias* di generazione delle funzioni booleane per ogni nodo pari a 0,79.

Inoltre, gli altri parametri di generazione sono stati fissati come segue:

- 4 nodi di input;

- 2 nodi di output;

- un *bias* di generazione delle funzioni booleane relative all'attivazione dell'output pari a 0.5 [4];

- 3 possibili *rewiring* dei nodi di input nella fase di adattamento all'ambiente.

3.6.1 Benchmark singola arena

I valori della distribuzione dei risultati sull'arena A, con il percorso integro, risultano essere estremamente più significativi rispetto all'arena B. Questo era già stato supposto in fase di creazione dell'arena.

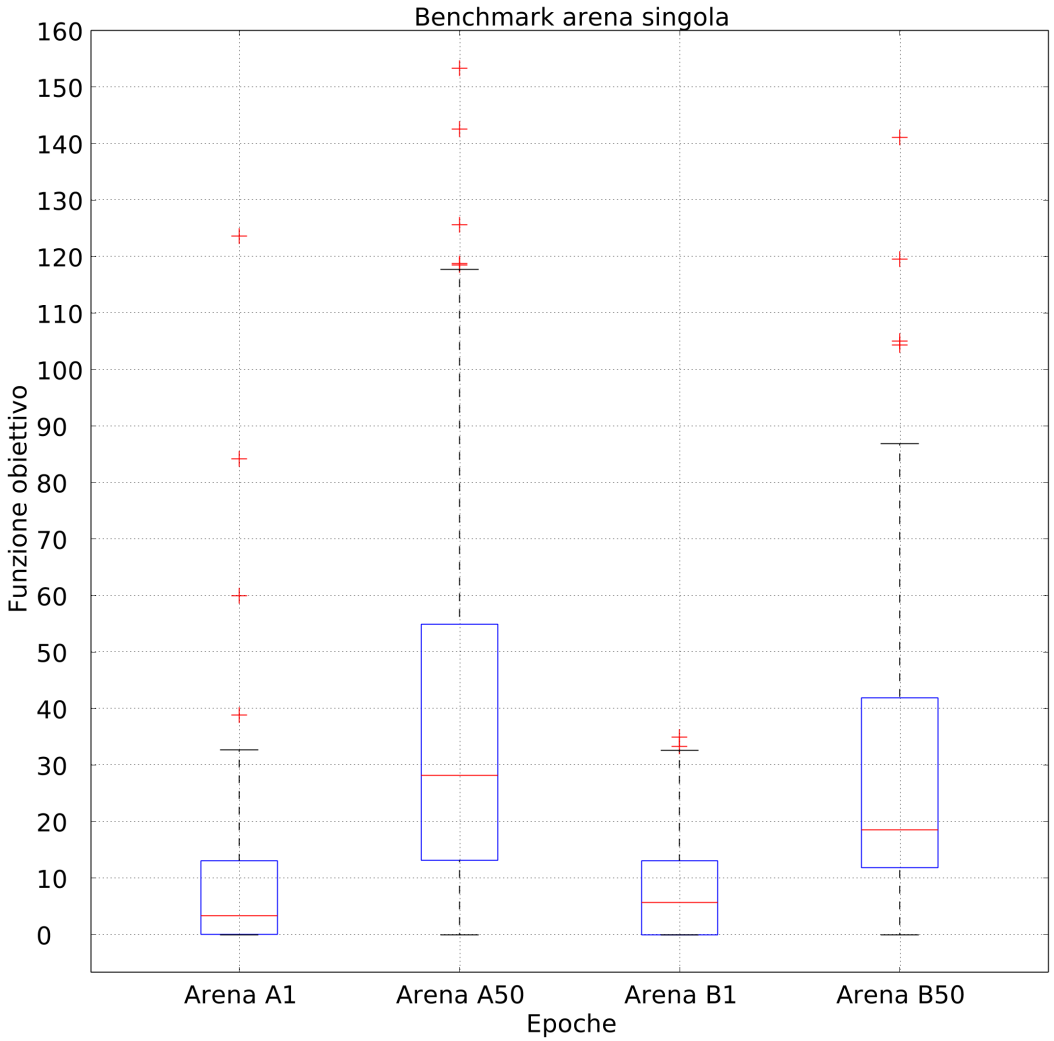
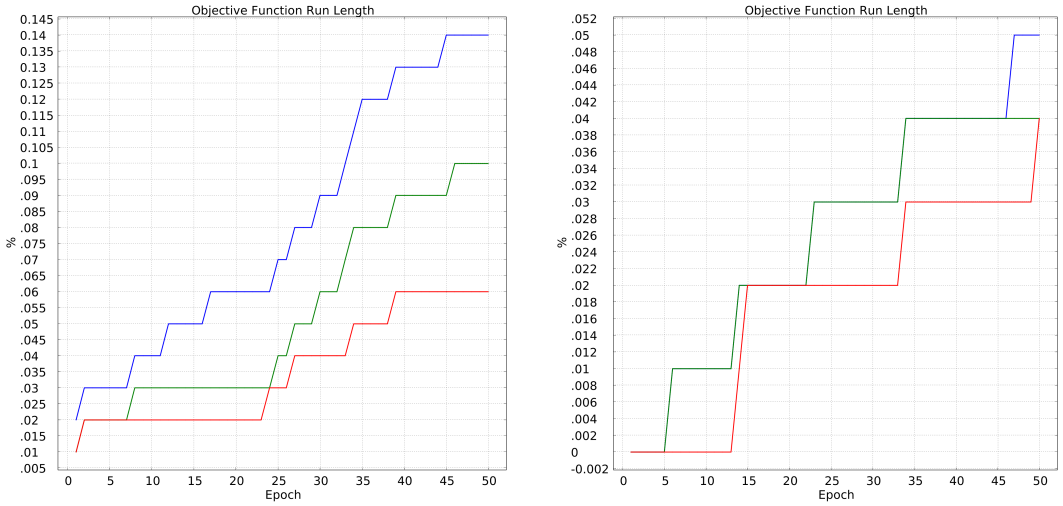


Figura 3.33: Nella metà sinistra (risp. destra), la distribuzione dei valori di funzione obiettivo di benchmark per la sola arena A (risp. B)

Nonostante gli alti valori raggiunti dalla funzione obiettivo, soltanto il 6% degli esemplari nella simulazione in ambiente A ha raggiunto il 70% della funzione obiettivo massima, mentre solo il 4% degli esemplari adattati nell'ambiente B ha raggiunto tale soglia.



(a) RLD su ambiente B

(b) RLD su ambiente B

Figura 3.34: *Run length distribution* degli ambienti di benchmark ad arena singola: le tracce blu, verdi e rosse rappresentano rispettivamente la soglia di calcolo posta al 50, 60 e 70 per cento del valore massimo raggiunto

3.6.2 Tuning singola arena

In entrambe le arene dedicate all'esperimento di *Navigazione su tracciato* l'empowerment non si è saturato come nelle altre casistiche, seppure raggiungendo valori estremamente alti. Questo ha portato ad un *cutoff* estremamente alto per entrambe le arene, con 18 epoche dedicate al preadattamento nel caso dell'ambiente A e 12 all'ambiente B.

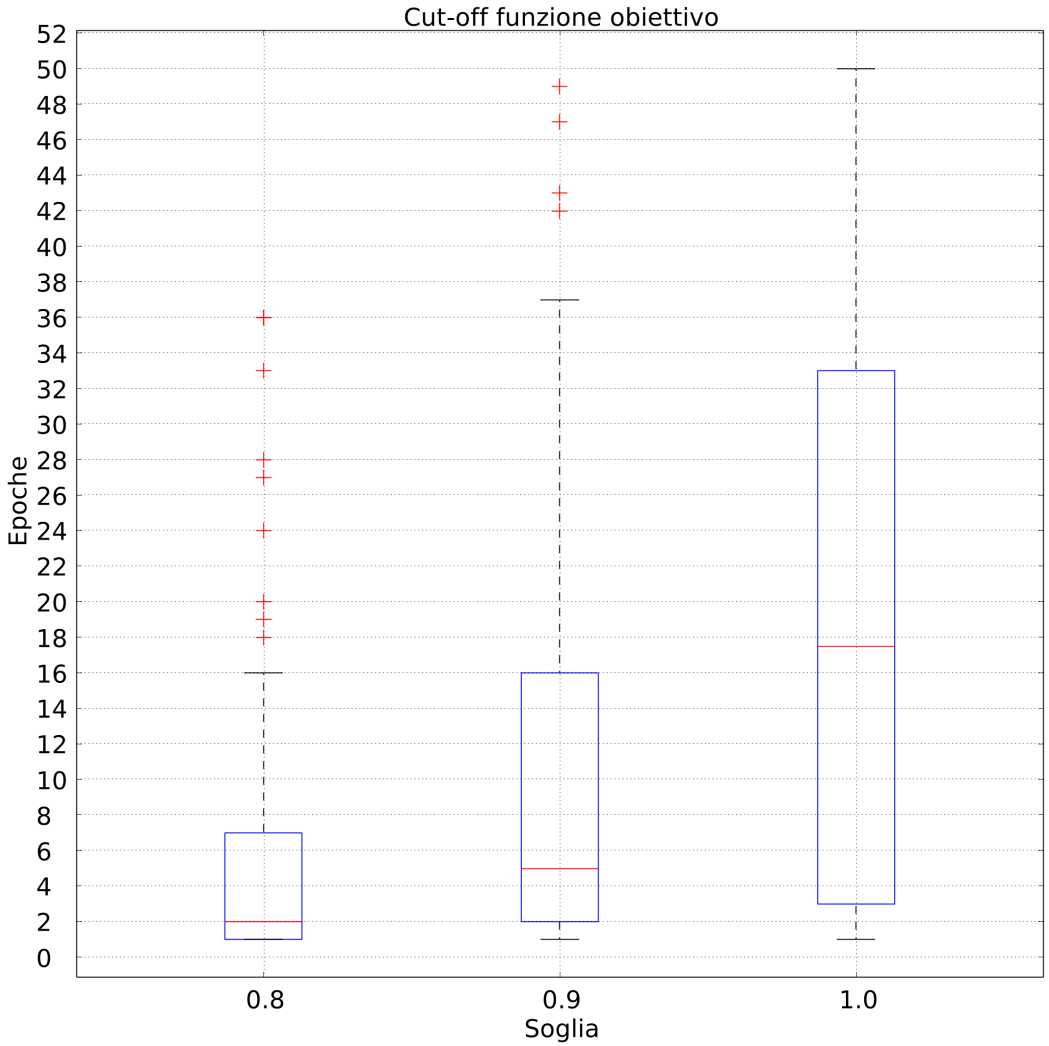


Figura 3.35: Soglie di cutoff su ambiente A

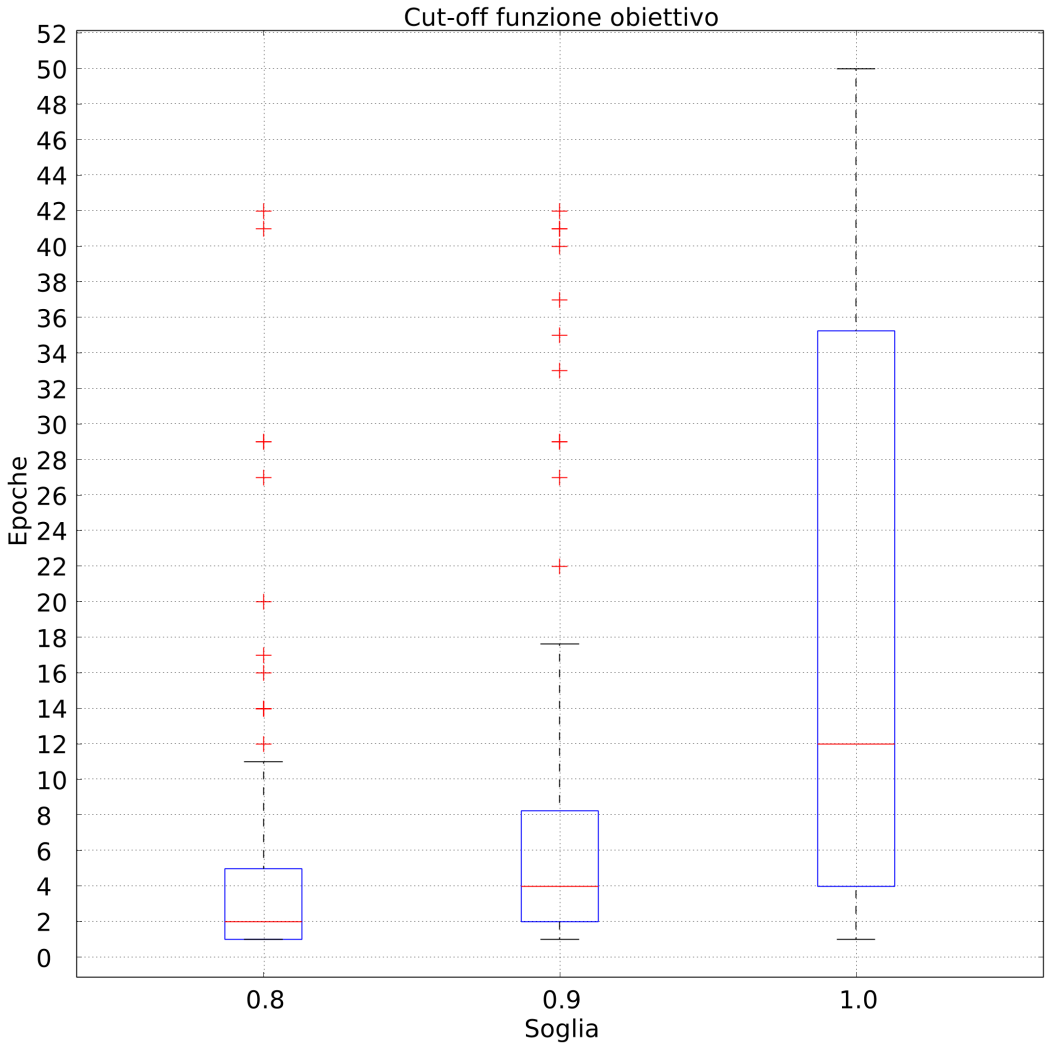


Figura 3.36: Soglie di cutoff su ambiente B

La distribuzione dei valori di funzione obiettivo raggiunte utilizzando il solo empowerment come funzione di adattamento risulta essere poco accentuata, limitandosi al valore nel terzo quartile di 30.

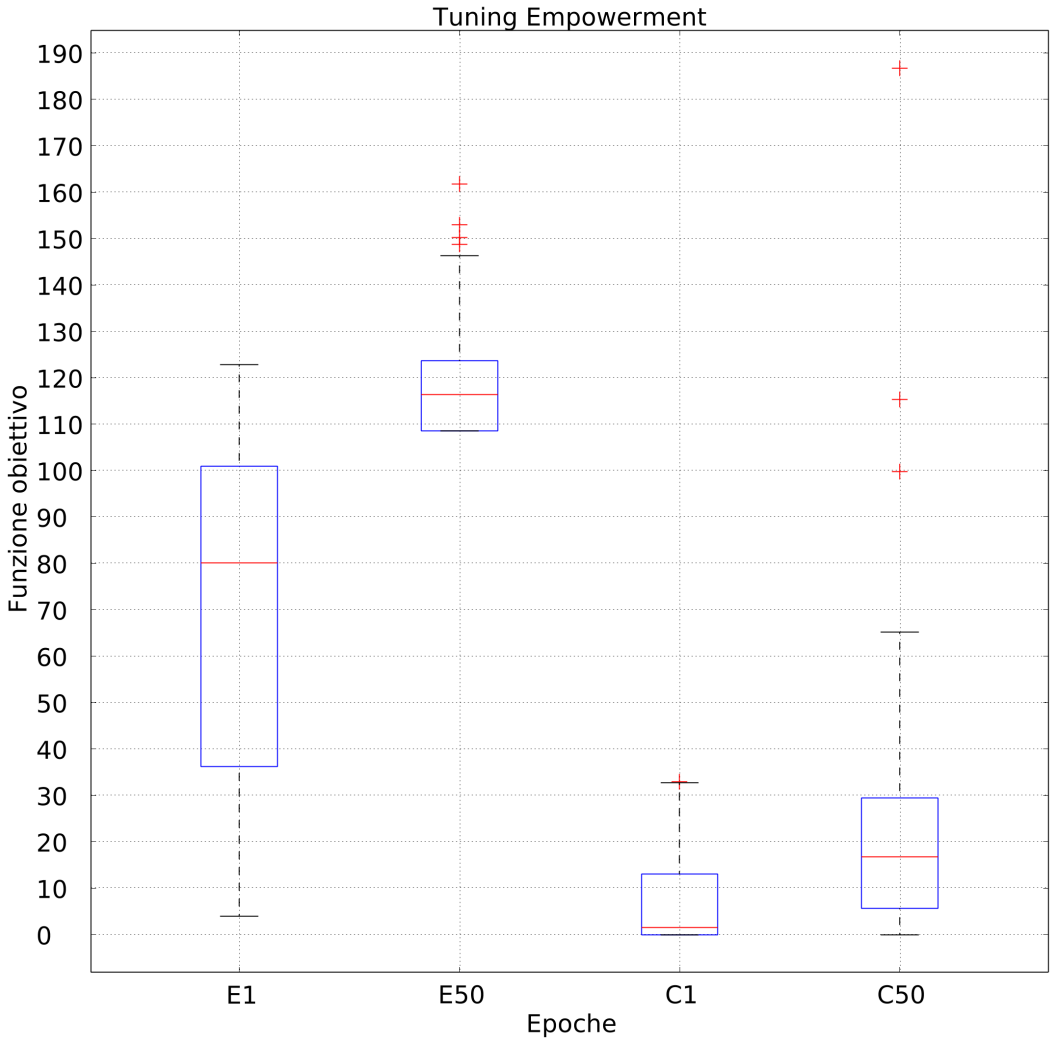


Figura 3.37: Distribuzioni ad epoca 1 e 50 delle funzioni di empowerment (sinistra) e obiettivo (destra) su ambiente A

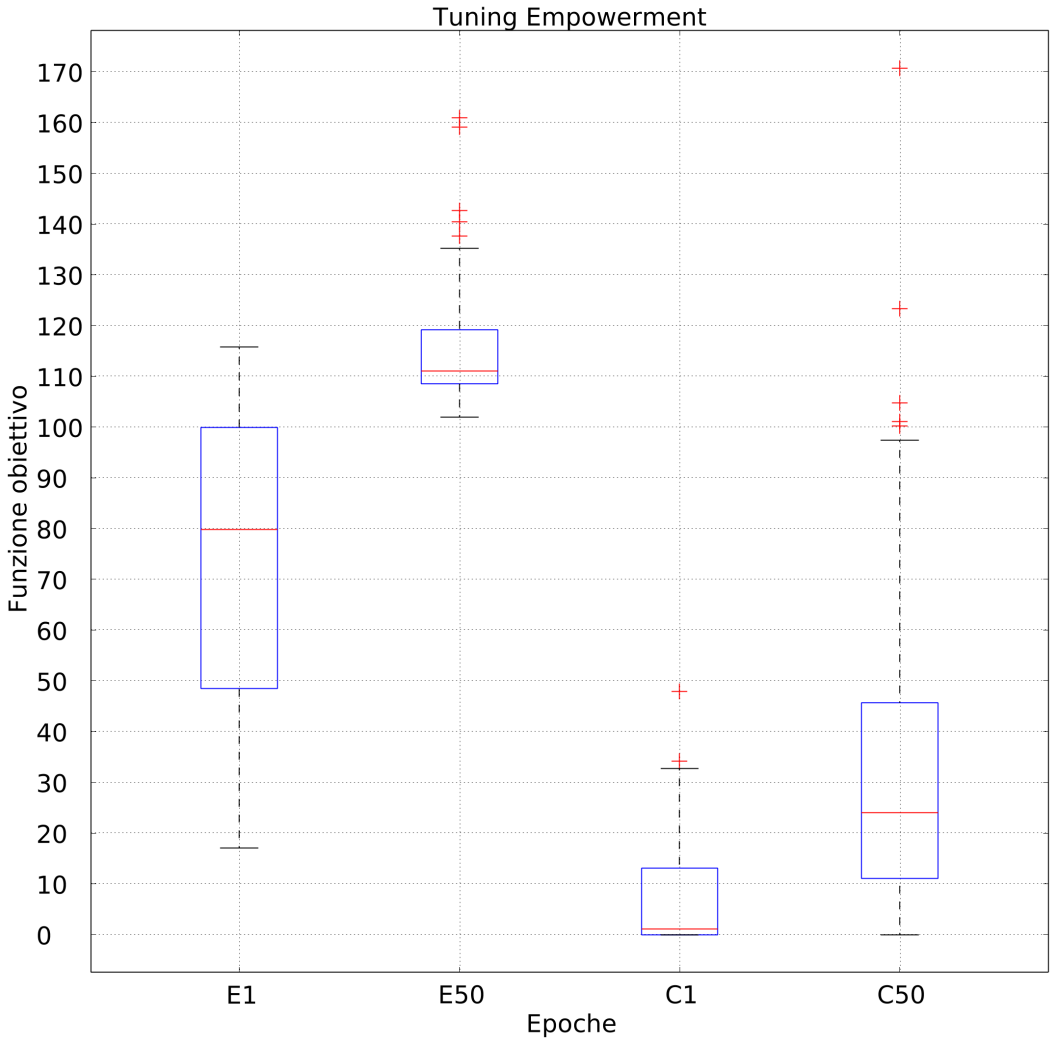


Figura 3.38: Distribuzioni ad epoca 1 e 50 delle funzioni di empowerment (sinistra) e obiettivo (destra) su ambiente B

3.6.3 Benchmark sperimentale

Il trasferimento del miglior esemplare dall'arena A all'arena B, e rispettivamente dall'arena B all'arena A ha mostrato un comportamento similare nella distribuzione dei valori della funzione obiettivo: in entrambe le casistiche si è avuto un leggero incremento nei valori massimi del terzo quartile, e

contemporaneamente della mediana.

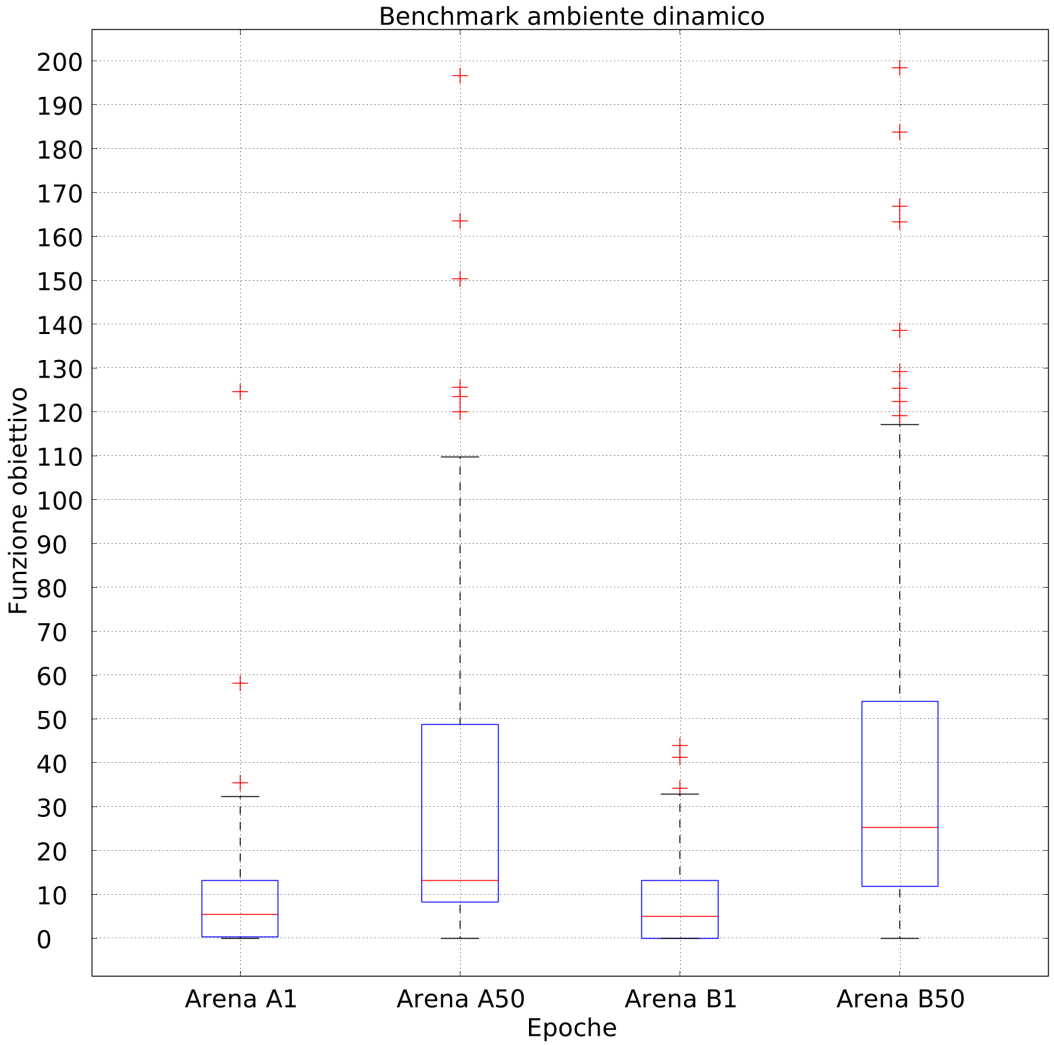


Figura 3.39: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena A, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena A nell'arena B

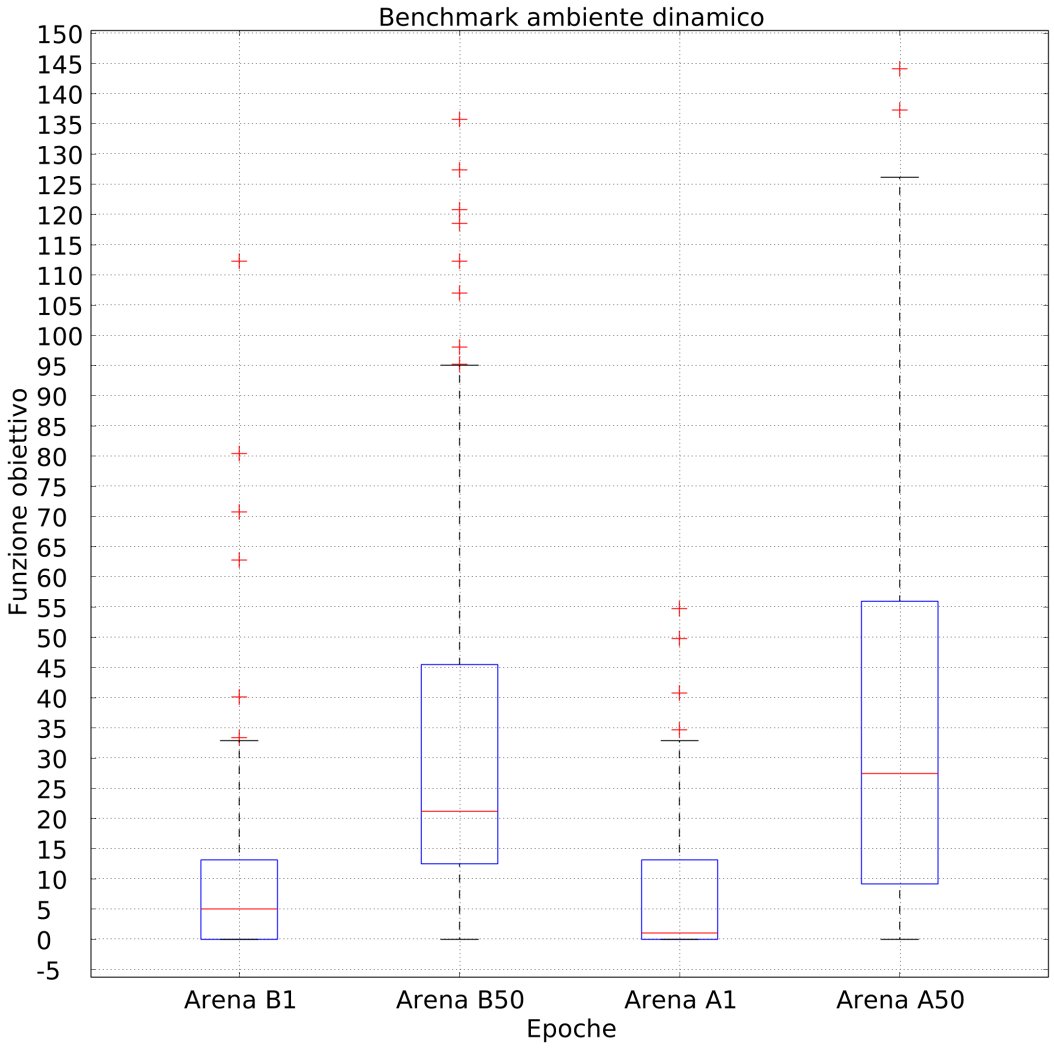


Figura 3.40: Nella metà sinistra, la distribuzione dei valori di funzione obiettivo all'interno dell'arena B, a destra la distribuzione raggiunta dopo aver iniettato il miglior esemplare dell'arena B nell'arena A

3.6.4 Sperimentazione

Il passaggio da ambiente A con preadattamento ad ambiente B ha portato ad un restringimento della distribuzione dei valori della funzione obiettivo, ed un innalzamente della mediana degli stessi, al momento del trasferimento nel

secondo ambiente.

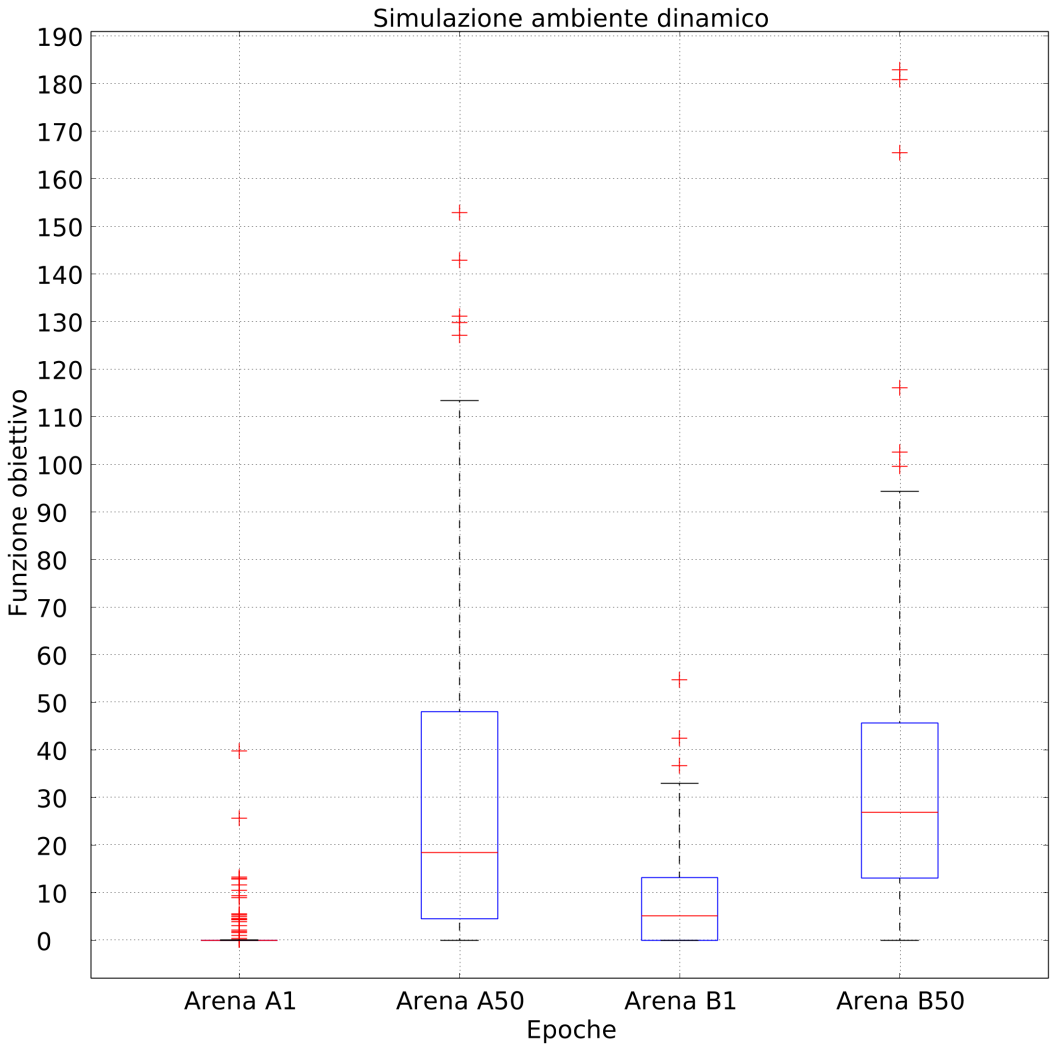


Figura 3.41: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena A, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena B

Effettuando il preadattamento nell'arena B, il trasferimento nell'arena A ha portato ad un miglioramento sia della distribuzione dei valori che della mediana.

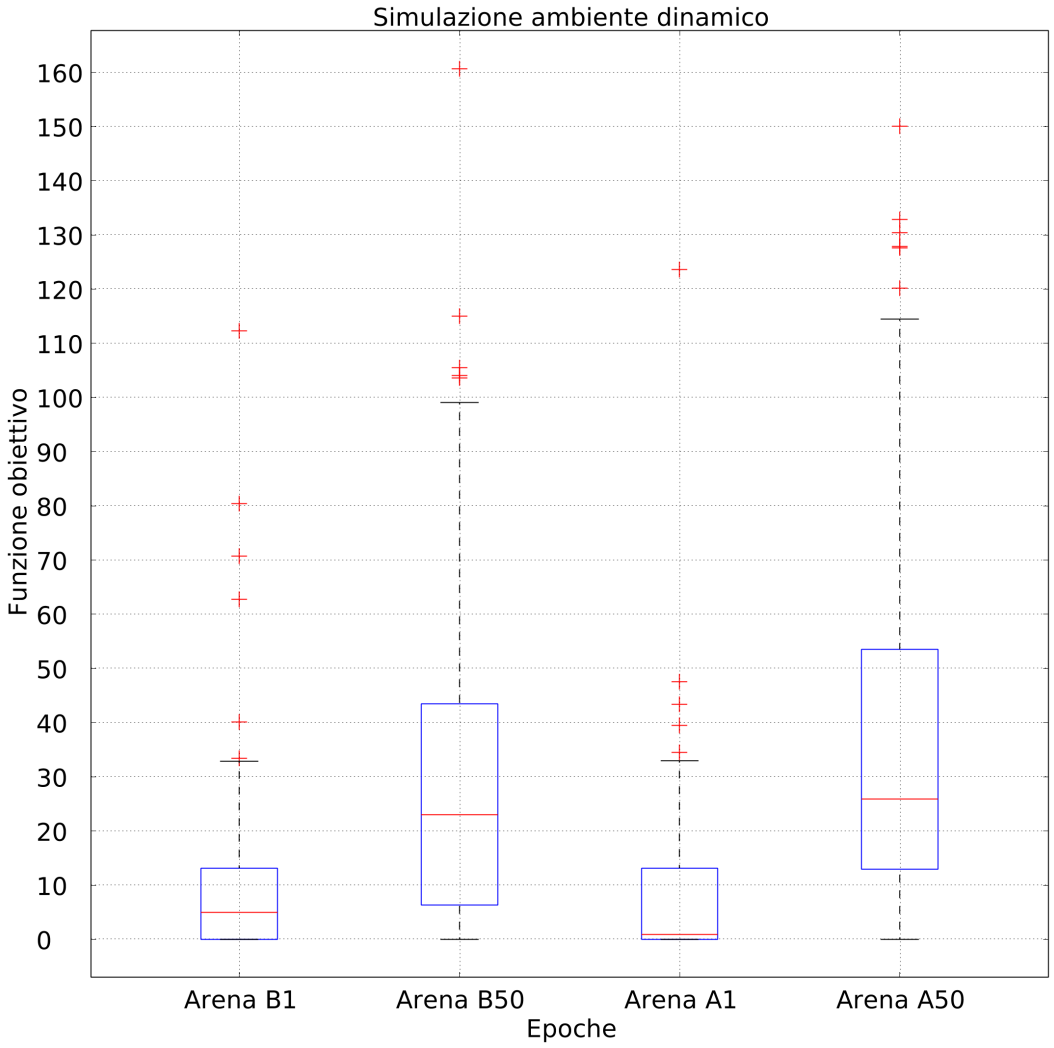


Figura 3.42: Nella metà sinistra, la distribuzione dei valori massimi di funzione obiettivo di una RBN con preadattamento su arena B, nella metà destra i risultati del miglior esemplare del primo ambiente trasferito nell'arena A

3.6.5 Risultati

Il confronto fra le distribuzioni di valori di benchmark e di sperimentazione mostra come i risultati sperimentali riescano al più a raggiungere i valori di controllo, ma mai superarli.

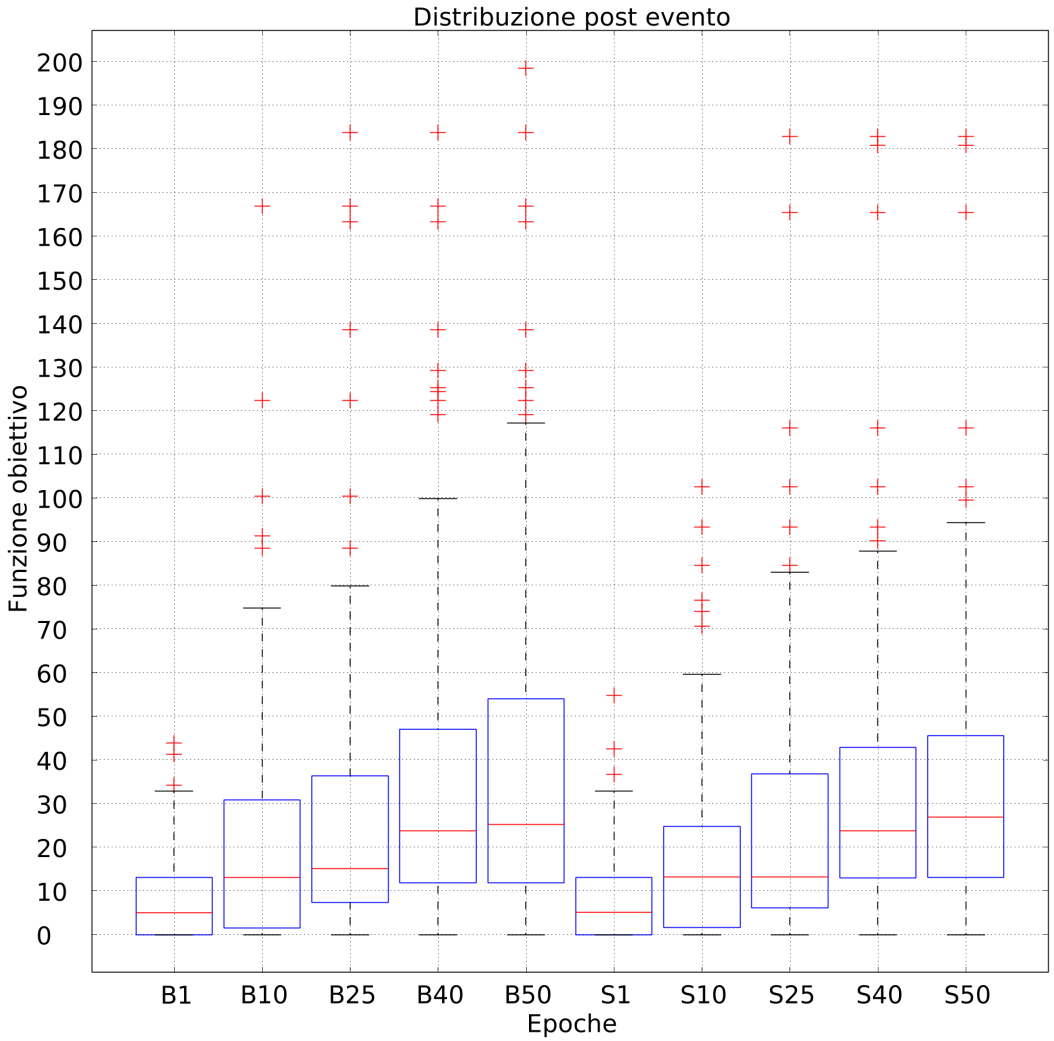


Figura 3.43: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente A come fase finale

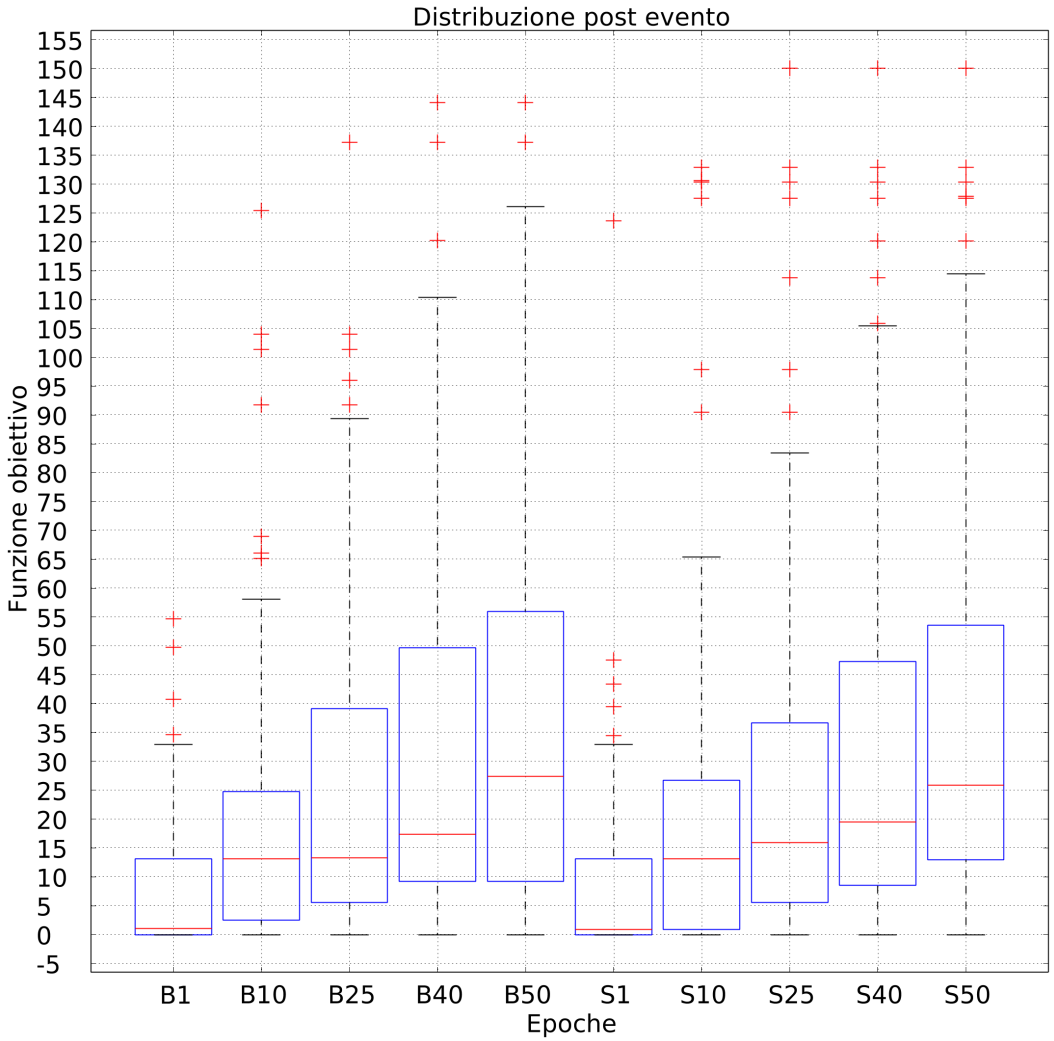
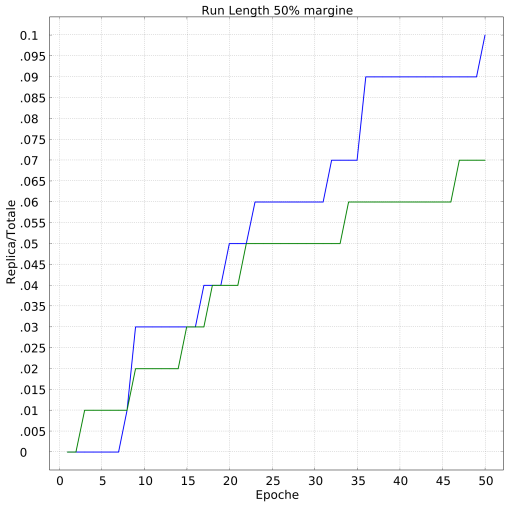
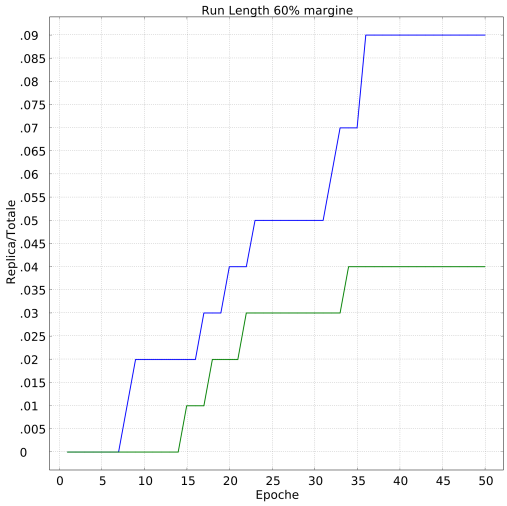


Figura 3.44: Campioni alle epoche 1, 10, 25, 40 e 50 della progressione di valori di funzione obiettivo massimizzata in benchmark (sinistra) ed in fase sperimentale (destra) su ambiente A come fase finale

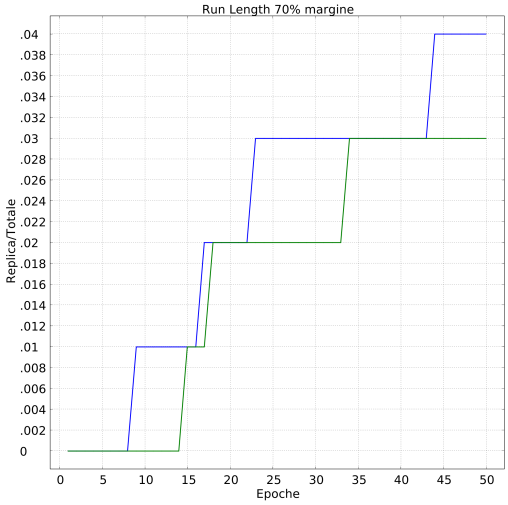
I *run length distribution* nei due ordinamenti di esecuzione delle arene mostrano come nell'ordinamento AB la velocità di massimizzazione della funzione obiettivo nelle varie soglie sia sempre maggiore per la fase di benchmark, e che valga il contrario per l'ordinamento BA, dove l'andamento della distribuzione di sperimentazione domina chiaramente l'andamento del benchmark.



(a) RLD tarate al 50%

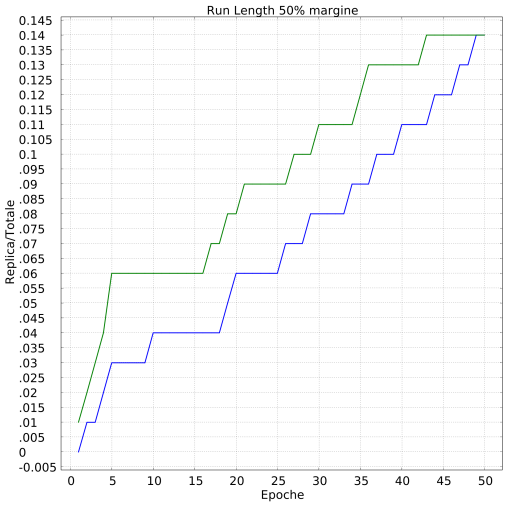


(b) RLD tarate al 60%

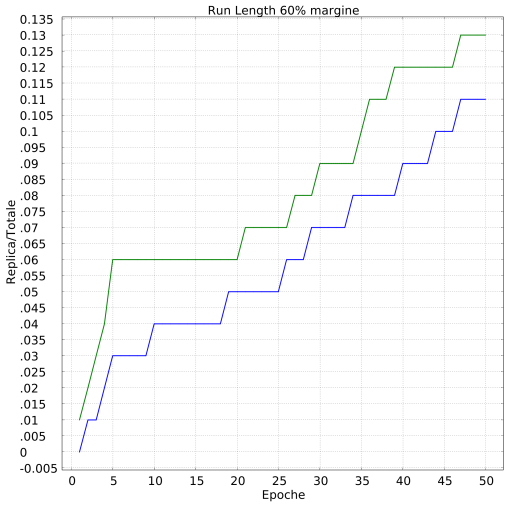


(c) RLD tarate al 70%

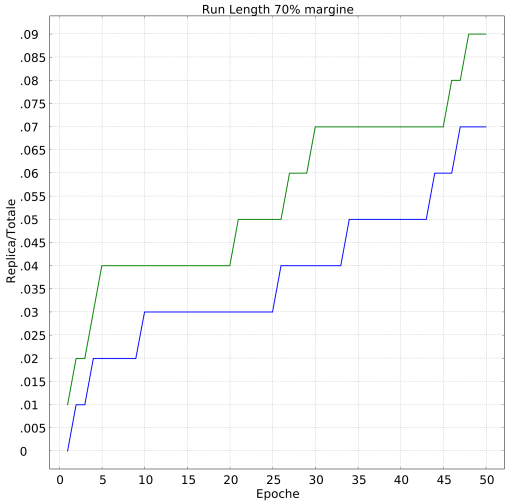
Figura 3.45: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale



(a) RLD tarate al 50%



(b) RLD tarate al 60%



(c) RLD tarate al 70%

Figura 3.46: *Run length distribution* dell'ambiente B quando usato come fase finale della simulazione: la traccia blu si intende relativa alla fase di benchmark, la traccia verde si intende relativa alla fase sperimentale

Su tutte le simulazioni viene effettuato il test di indipendenza statistica di MannWhitney. Il valore di α per rifiutare l'ipotesi nulla si considera fissato a 0.05. Con *init* e *max* ci si riferisce rispettivamente alla distribuzione dei valori

ad epoca 0 e ai massimi valori raggiunti a fine simulazione. I termini affiancati da una b o una s rimandano rispettivamente a dei risultati di benchmark e sperimentazione.

I risultati mostrano come nella fase di benchmark ad arena singola si possa rifiutare l'ipotesi nulla per il processo di adattamento su entrambe le arene, anche possibile confrontando la distribuzione dei valori massimizzati raggiunti nelle due arene singolarmente.

serie 1	serie 2	*p-value*	H_0
init	max	$4.2693229111277406E-14$	Rifiutata
init	max	$1.2676753562066848E-10$	Rifiutata
max A	max B	0.0271869160699283	Rifiutata

Effettuando i test sui valori della funzione obiettivo associata all'empowerment nella fase di tuning, si possono rifiutare entrambe le ipotesi nulle per i valori iniziali e finali

serie 1	serie 2	*p-value*	H_0
init	max	$1.8030191112139812E-10$	Rifiutata
init	max	$1.5228972483761064E-13$	Rifiutata

Nella fase di benchmark ad arena doppia si può rifiutare l'ipotesi nulla per entrambi i processi di adattamento, considerando come *init* la distribuzione dei valori alla prima epoca della prima arena, come *max* la distribuzione dei valori massimi al termine della simulazione nel secondo ambiente. Effettuando il test statistico fra le distribuzioni dei valori risultanti alla fine delle simulazioni, non è possibile rifiutare l'ipotesi nulla.

serie 1	serie 2	*p-value*	H_0
init A	max B	$1.5245247402651307E-10$	Rifiutata
init B	max A	$5.427840743272315E-10$	Rifiutata
maxed AB	maxed BA	0.97563450986244	Non rifiutata

Nella fase di simulazione sperimentale risulta confermato il rifiuto delle ipotesi nulle in entrambi i processi di adattamento, dove con *init A* (risp. *init B*) e *max A* (risp. *max B*) si intendono la prima epoca nel contesto della prima arena dell'esperimento e l'ultima epoca nella seconda fase. Non è possibile rifiutare l'ipotesi nulla nel caso del confronto fra le distribuzioni di valori delle funzioni obiettivo massimizzate nelle due fasi finali dei due ordini dell'esperimento, indicati in tabella con *max AB* e *max BA*.

serie 1	serie 2	*p-value*	H_0
init A	max B	$8.72519552293715E-27$	Rifiutata
init B	max A	$7.373662575559129E-11$	Rifiutata
max AB	max BA	0.9805065169785967	Non rifiutata

Effettuando il test statistico sulle distribuzioni di valori massimizzate nelle fasi di benchmark e sperimentazione, non si può rifiutare l'ipotesi nulla in nessuno degli ordini di adattamento.

serie 1	serie 2	*p-value*	H_0
max AB(b)	max AB(s)	0.9425382646537455	Non rifiutata
max BA(b)	max BA(s)	0.8815167657646835	Non rifiutata

Conclusioni e sviluppi futuri

3.7 Conclusioni

I risultati raggiunti nelle simulazioni effettuate nei tre diversi esperimenti hanno risposto in buona parte alle domande formulate in sezione 3.2.

Emerge chiaramente che l'empowerment abbia una valenza come motore del processo adattativo: in tutti e tre gli esperimenti, seppure non riuscendo ad eguagliare le distribuzioni di valori ottenuti da un adattamento guidato da funzione obiettivo specifica, sono stati raggiunti picchi notevoli di aderenza allo svolgimento del task, considerando che i robot non non ne conoscevano alcun precetto se non quali strutture sensoriali ne sono state coinvolte. In particolare è rilevante notare come, nel confronto dei risultati a livello di aderenza del singolo task, i migliori siano stati raggiunti nelle arene ritenute più complesse, in linea con alcune impressioni avute dagli autori del concetto di empowerment[7]. Il mancato raggiungimento di tale obiettivo nel tuning ad arena singola dell'esperimento di Ricerca spaziale assistita potrebbe essere imputabile alla metodologia con la quale la maggiore difficoltà è stata imposta nel sistema: se negli altri due esperimenti questo si è ottenuto tramite modificazioni che rendessero meno agibile lo svolgimento del task, in questo la complessità è stata ottenuta con l'aggiunta di una forte condizione di rumore

positivo. Per un sistema basato su empowerment, la presenza di poche, ma importanti fonti di attivazione dei sensori come le grosse macchie per terra che caratterizzano la seconda arena dell'esperimento di Ricerca spaziale assistita ha lo stesso effetto di disturbo [5] teorizzato per lo svolgimento dello stesso esperimento guidato da funzione obiettivo specifica.

In relazione all'ipotesi che si chiedeva se un adattamento guidato da empowerment potesse essere quantitativamente più efficace di un adattamento guidato da funzione obiettivo, la risposta ottenuta dai risultati è stata nettamente negativa in ogni casistica delle simulazioni ad arena singola. L'unica eccezione è stata riscontrata nella seconda arena dell'esperimento Navigazione su tracciato, di nuovo l'arena ritenuta più ostica della coppia, in cui i risultati ottenuti dall'adattamento secondo empowerment hanno superato quelli ottenuti da funzione obiettivo. Il risultato sembra essere in linea con le osservazioni delineate dagli autori dell'empowerment[5], secondo i quali una fase sensoriale caratterizzata da una rapida oscillazione di valori possa risultare benefico nell'applicazione della funzione.

L'analisi dell'applicazione del concetto di empowerment in un contesto più complesso delle simulazioni ad arena singola, facendo quindi un passo verso l'utilizzo in contesti robotici concreti, ha portato a dei risultati quantitativi contrastanti, in linea con le conclusioni già delineate, che restringono le possibili strade di ricerca futura in delle direzioni molto chiare. L'aspetto qualitativo del trasporto di una rete che ha già completato il proprio ciclo di adattamento con applicazione preventiva di empowerment in un ambiente ancora non conosciuto viene analizzato maggiormente nelle prime epoche, per valutare l'approccio al nuovo ambiente, e nelle ultime, per valutare i risultati complessivi raggiunti

dall'adattamento. In nessuna delle configurazioni sperimentali in cui il trasferimento dell'esemplare pre-adattato viene effettuato verso l'arena considerata più ostica si è ottenuto, nè nelle prime fasi dell'approccio alla nuova arena, nè nelle rilevazioni finali, un miglioramento delle prestazioni rispetto alle simulazioni di benchmark, suggerendo addirittura che la fase di pre-adattamento sia deleteria al raggiungimento di buoni livelli di prestazione in seguito ad un turbamento dell'ambiente. Tuttavia, quando il pre-adattamento viene effettuato nelle arene più complicate per poi trasportare il migliore esemplare a completare la sperimentazione nelle arene più semplici si ha avuto un incremento percentuale nelle prestazioni, sia nelle misurazioni a fronte del cambiamento che in quelle effettuate a fine simulazione. Tramite questi risultati, assolutamente in linea con le osservazioni fatte nelle simulazioni ad arena singola, si può desumere come la metodologia utilizzata di massimizzazione dell'empowerment abbia una rilevanza in un contesto di cambio ambientale quando utilizzato in ambienti complessi.

3.8 Sviluppi futuri

Sulla base dei risultati raggiunti è possibile immaginare una serie di sperimentazioni basate sull'espansione della complessità ambientale alla quale sono sottoposti i robot, sia in contesti ad arena singola che con un'espansione della linea temporale formata dagli ambienti multipli.

La migliore risposta delle reti booleane critiche *empowered* quando sottoposte ad una elevata complessità potrebbe portare ad una generalizzazione inversa, per la quale risolvere in maniera euristica un problema complesso potrebbe portare ad una soluzione più che soddisfacente delle sue sottoparti, in una sorta di applicazione automatizzata del concetto di *divide et impera*.

Inoltre, è possibile utilizzare altre forme di applicazione dell'empowerment all'adattamento, sia in ambito strutturale che sperimentale. Questa trattazione, nel suo approccio pionieristico dell'argomento, ne ha incrociati solo alcuni.

Da un punto di vista strumentale, il design modulare e l'approccio a sussunzione sul quale si è basato lo sviluppo del framework sperimentale pongono le basi per una facile espansione delle funzionalità e delle possibilità applicative nel dominio delle reti booleane.

Ringraziamenti

Sai chi sei.

Grazie per fare parte della mia vita.

Bibliografia

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

[2] Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

[3] Richard Blahut. Computation of channel capacity and rate-distortion functions. *IEEE transactions on Information Theory*, 18(4):460–473, 1972.

[4] Michele Braccini, Andrea Roli, and Stuart Kauffman. Online adaptation in robots as biological development provides phenotypic plasticity. 06 2020.

[5] Nicola Catenacci Volpi and Daniel Polani. Goal-directed empowerment: Combining intrinsic motivation and task-oriented behaviour. *IEEE Transactions on Cognitive and Developmental Systems*, PP:1–1, 12 2020.

[6] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[7] A.S. Klyubin, D. Polani, and C.L. Nehaniv. Empowerment: a universal agent-centric measure of control. 1:128–135 Vol.1, 2005.

[8] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[9] Andrea Roli and Michele Braccini. Attractor landscape: A bridge between robotics and synthetic biology. *Complex Systems*, 27:229–248, 10 2018.

[10] Andrea Roli, Michele Braccini, and Edoardo Barbieri. Adattamento online di robot controllati da reti booleane.

[11] Andrea Roli, Marco Villani, Alessandro Filisetti, and Roberto Serra. Dynamical criticality: overview and open questions, 2016.

Appendice A

Design controller Applicare un design modulare al controller utilizzato per gestire l'adattamento di un'intera replica si è rivelato fondamentale per gestire la crescente complessità gestionale ottenuta dalla possibilità di poter trasferire il doppio delle informazioni finora ritenute possibili e la necessità di soddisfare due progetti di tesi che seguono un precetto comune, ma dal flusso esecutivo molto differente.

Partendo dall'interfaccia che ARGoS pretende rispettata per accettare un controller, si sono delineate delle librerie che inglobassero il controllo delle singole responsabilità al contorno:

- `BooleanNetwork` è il modulo che gestisce il comportamento interno di qualsiasi aspetto di un rete booleana. Fornisce l'intera gamma di metodi accessori e modificatori tipici del concetto di classe (omessi dal diagramma), seppure Lua non possieda un concetto di modificatori di accesso per i campi di un modulo, e due metodi che incapsulano la generazione di una possibile successiva fase di adattamento;

- `BooleanNetworkBuilder` è il *builder* che permette la creazione delle reti booleane, partendo da dei parametri di configurazione o da uno stato pre-esistente che si vuole iniettare all'interno di una rete;

- `Serializer` è il modulo che gestisce la trasformazione dei risultati ottenuti nel formato desiderato sulla base dei parametri forniti, riassunti in *network* ed *environment* per facilità di lettura;

- `Loader` è il modulo che gestisce il caricamento di valori esterni al controller con i quali poter creare reti booleane a partire da parametri di configurazione o valori preesistenti. Collabora strettamente con il `BooleanNetworkBuilder`.

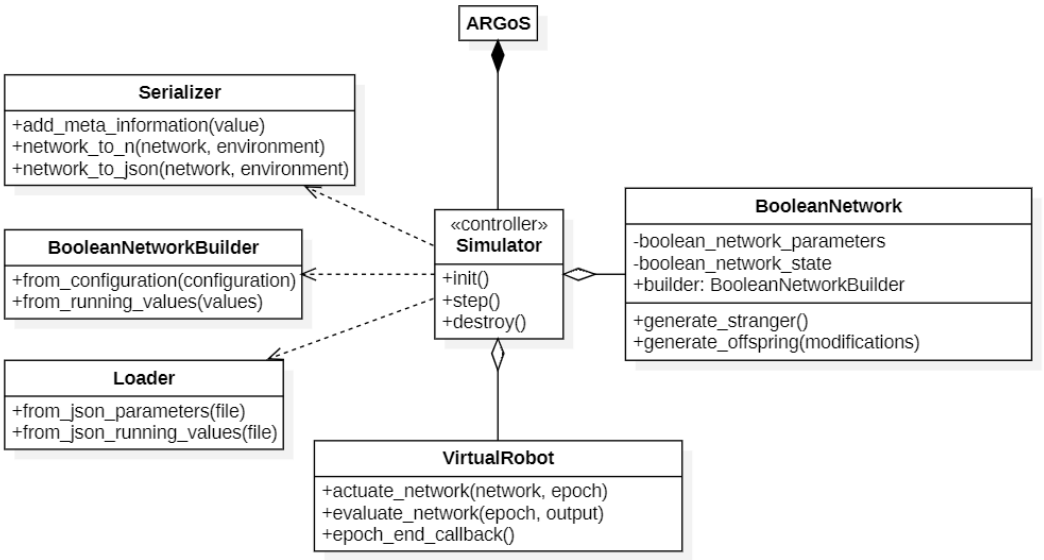


Figura 47: Il modello logico del controller lato ARGoS del sistema

Un'altra priorità è stata quella di estrapolare il concetto di funzionamento effettivo del robot al di fuori del controller, così che le responsabilità di quest'ultimo restassero ad un livello gestionale e soprattutto invariate a fronte dell'introduzione di diverse modalità esecutive di basso livello. Questo è stato realizzato attraverso il modulo `VirtualRobot`, che in quanto *adapter* di tutti i vari possibili robot contiene i metodi rilevanti per gestirne uno, ovvero:

- *actuate_network(...)* - utilizzato per effettuare un passo del *loop* sensori-motorio del robot all'interno dell'ambiente;

- *evaluate_network(...)* - utilizzato per conoscere i valori delle diverse funzioni obiettivo calcolate dal robot effettivo;

- *epoch_end_callback()* - utilizzata dal controller per informare il robot effettivo che è terminata un'epoca.

Il singolo punto di accesso alla fase di controllo di adattamento del robot con astrazione sull'effettiva istanza robotica ha quindi permesso di utilizzare un unico passo simulativo, riassunto nello pseudocodice in listato 1

```
function step()
 if (alive) then
   last_output = virtual.actuate_network_step()
   objective_differential, contestant_differential =
       virtual.evaluate_network()
   objective += objective_differential
   contestant += contestant_differential
   _step = _step + 1
 else
   normalized_objective = normalize(objective)
   normalized_contestant = normalize(contestant)
    if (normalized_objective >= best_objective) then
     best = current
     best_objective = normalized_objective
 end

   print(
    serializer.network_to_n(
     best,
     best_objective,
     normalized_objective,
     normalized_contestant,
     {environment}
    )
   )

  current = best:generate_offspring(MAX_INPUT_REWIRES)
  _epoch = _epoch + 1
  reset_epoch_environment()
 end
end
```

Listato 1: Scheletro implementativo in pseudocodice del metodo di *step* lato ARGoS

Ecosistema sensori Se dal lato controller è stato creato un adattamento logico del concetto di istanza robotica, dal lato sensoriale è stato realizzato un ecosistema di moduli che incapsulano il controllo della moltitudine di valori contenuti nel modulo ARGoS *robot*, unica interfaccia al *footbot* simulato nel sistema e ai valori che trasporta. Ogni modulo realizzato incapsula le capacità di un singolo sensore di quelli utilizzati nei vari esperimenti, fornendo un accesso regolato ai valori delle percezioni del robot, creando anche delle versioni logiche dei sensori che aggregano la moltitudine di letture effettive in formati convenienti alle simulazioni complesse e alle necessità dei robot effettivi implementati.

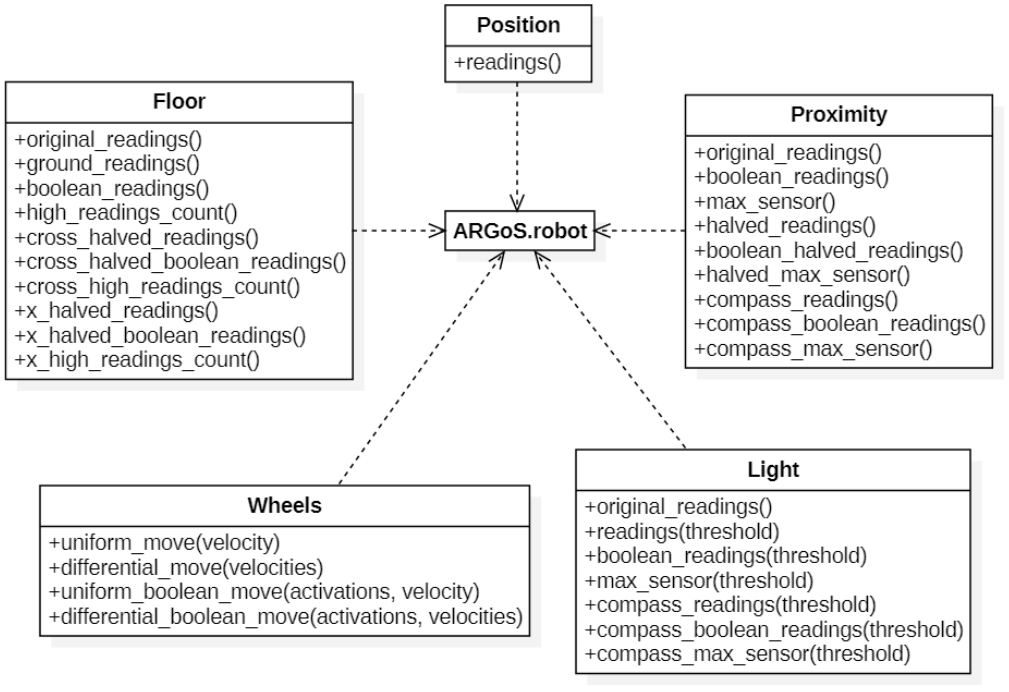
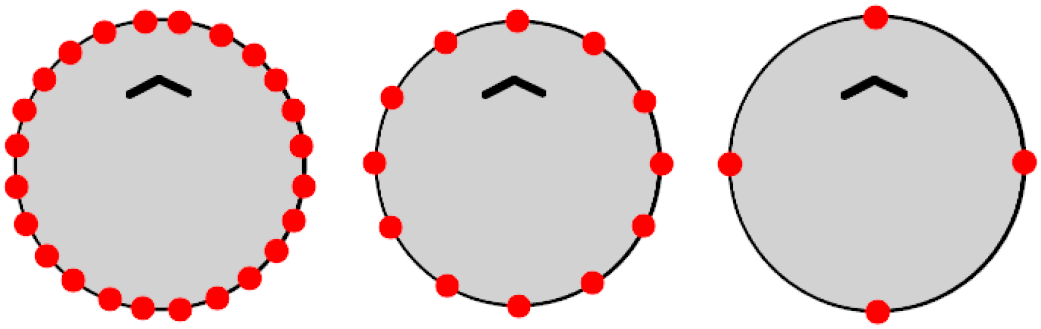


Figura 48: L'ecosistema di sensori logici sul quale si basa la sussunzione del sotto-sistema lato ARGoS

Il sistema di percezione luminoso montato a bordo del footbot simulato

conta 24 sensori, disposti sulla circonferenza del robot. Tramite il modulo `Light` viene fornita sia la lettura dei 24 sensori originali che una versione ridotta a 12, in cui i sensori vengono accoppiati fornendo una lettura media dei valori, e una versione "a bussola", in cui i valori ritornati sono soltanto 4



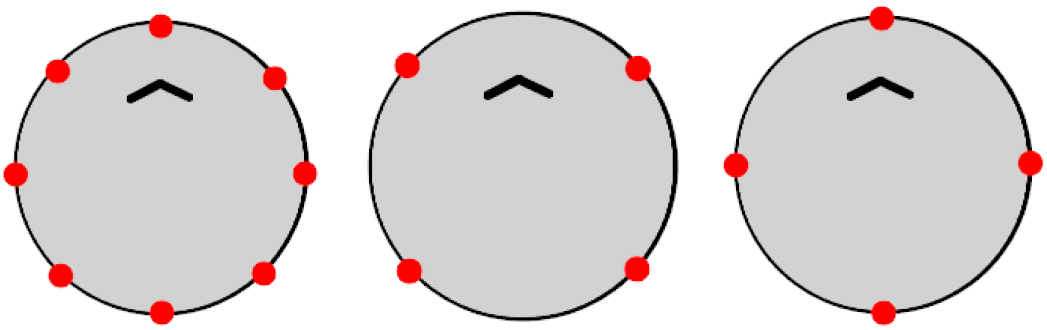
(a) Disposizione dei sensori di luminosità nella configurazione originale

(b) Disposizione dei sensori di luminosità nella configurazione "dimezzata"

(c) Disposizione dei sensori di luminosità nella configurazione "a bussola"

Figura 49: Configurazioni disponibili del sistema di percezione di luminosità

Per il sensore di terreno, incapsulato nel modulo `Floor`, nella configurazione originale conta 8 sensori. Di questo, oltre alla lettura dei valori originali, sono state realizzate due versioni dimezzate, in cui 4 sensori occupano le posizioni dei punti cardinali principali in una, le posizioni dei punti cardinali secondari nell'altra.



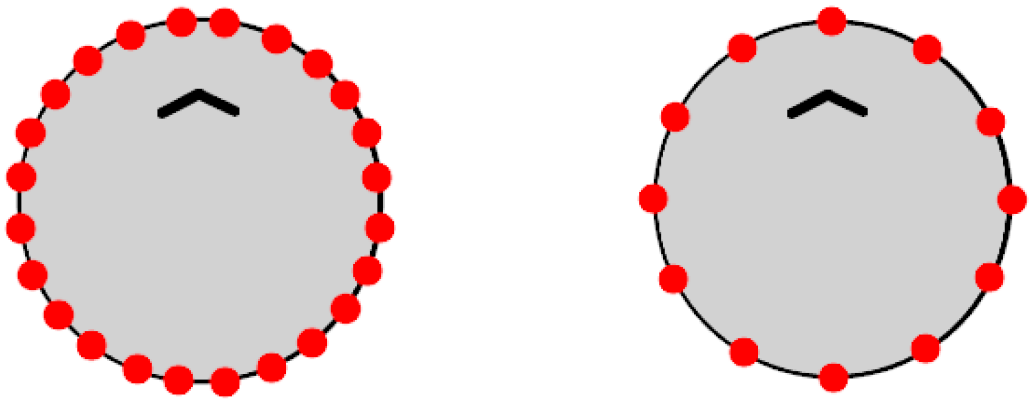
(a) Disposizioni dei sensori di terreno nella configurazione originale

(b) Disposizioni dei sensori di terreno nella configurazione a “x”

(c) Disposizioni dei sensori di terreno nella configurazione a “bussola”

Figura 50: Configurazioni disponibili del sistema di percezione di terreno

Il sensore di prossimità è stato astratto nel modulo `Proximity`, e realizzato nella forma originale e nella forma dimezzata, con 24 e 12 sensori disponibili



(a) Disposizioni dei sensori di prossimità nella configurazione originale

(b) Disposizioni dei sensori di prossimità nella configurazione “dimezzata”

Figura 51: Configurazioni disponibili del sistema di percezione di prossimità

Il modulo `Wheels` astrae il controllo sulle ruote del footbot, permettendo

diverse modalità di guida del robot a seconda delle necessità esplorative.

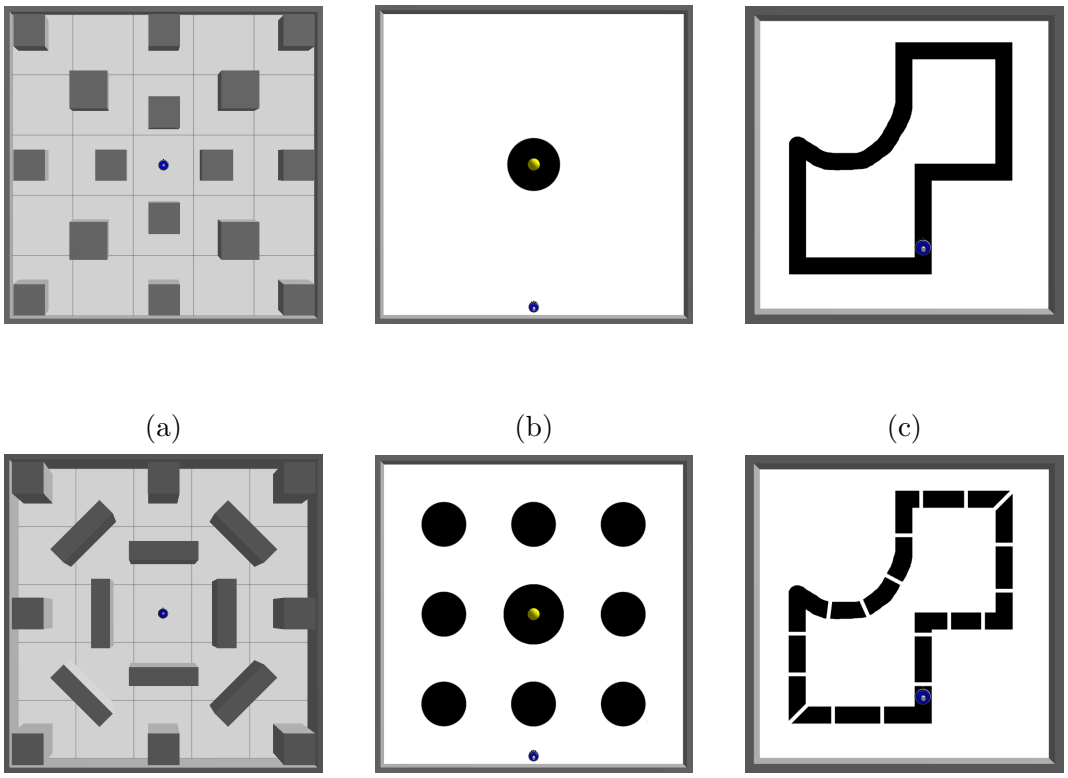
Il modulo `Position` pulisce l'output del sistema di GPS interno del footbot, riducendo i complicati output del simulatore ad un array da tre elementi numerici, rispettivamente coordinate rispetto agli assi x e y e rotazione rispetto all'asse z.

Appendice B

In questa appendice si mostrano i risultati effettivi a livello di capacità esplorativa raggiunti da alcuni dei più efficaci adattamenti nelle varie arene sottoposte a sperimentazione. I risultati che si propongono provengono esclusivamente dalle simulazioni di benchmark e tuning su arena singola[1]: da un punto così elevato di visualizzazione del comportamento non ha senso analizzare tutte le differenti configurazioni del sistema, in quanto non sarebbe possibile concretizzare alcuna osservazione dal solo comportamento del robot. È comunque interessante dare un volto ai diagrammi a scatole e baffi visti nel resto dell'elaborato, e visualizzare l'effettiva applicabilità dell'empowerment in un contesto di adattamento online, confrontato al "semplice" adattamento via funzione obiettivo.

In ogni grafico, ogni colore indica un'epoca particolare. Nonostante non sia di alcun valore analitico specialmente al crescere del numero di epoche, è possibile intuire un senso di stratificazione nei percorsi, considerando che le epoche vengono disegnate in ordine e che quindi le più "vicine" a noi sono anche le più recenti.

[1]Si ricorda che le simulazioni di benchmark sono guidate dalla massimizzazione di una funzione obiettivo specifica al task da portare a compimento; le simulazioni di tuning sono guidate dalla massimizzazione dell'empowerment



(a) (b) (c)

(d) (e) (f)

Figura 52: I sei ambienti coinvolti nelle fasi sperimentali

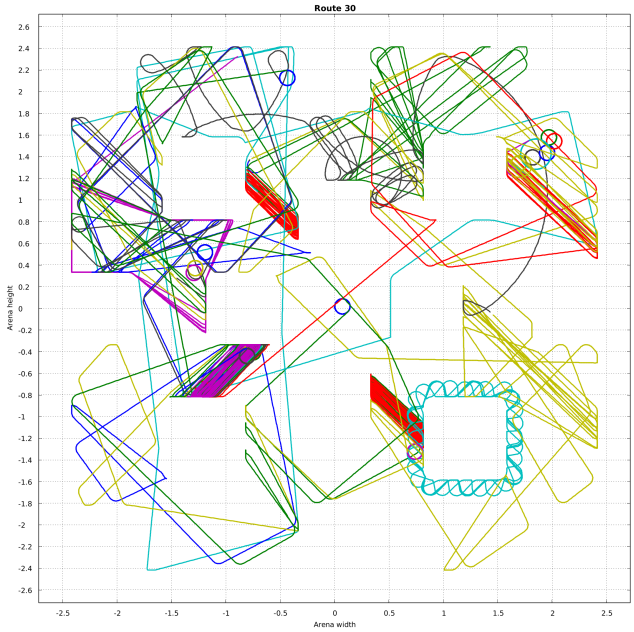


Figura 53: Benchmark su arena 6.a nell'esperimento *Navigazione fra ostacoli*

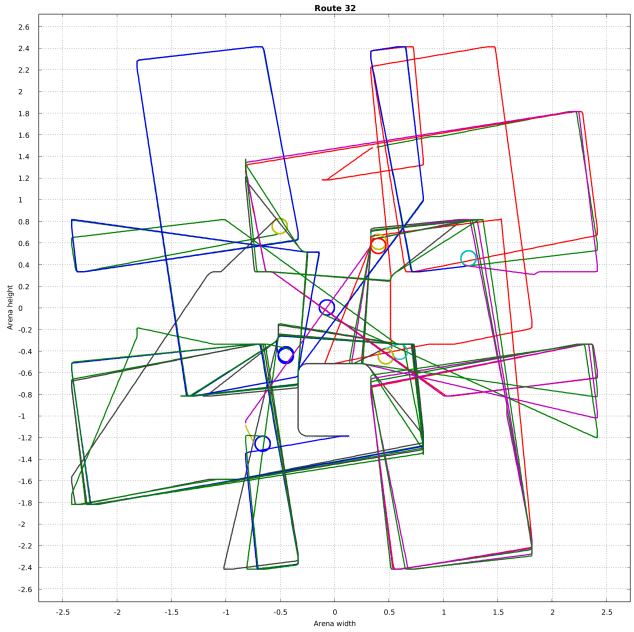


Figura 54: Tuning su arena 6.a nell'esperimento *Navigazione fra ostacoli*

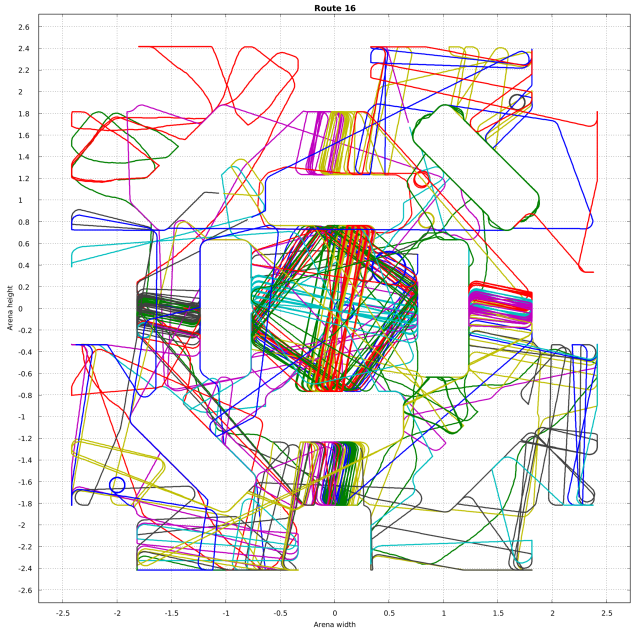


Figura 55: Benchmark su arena 6.d nell'esperimento *Navigazione fra ostacoli*

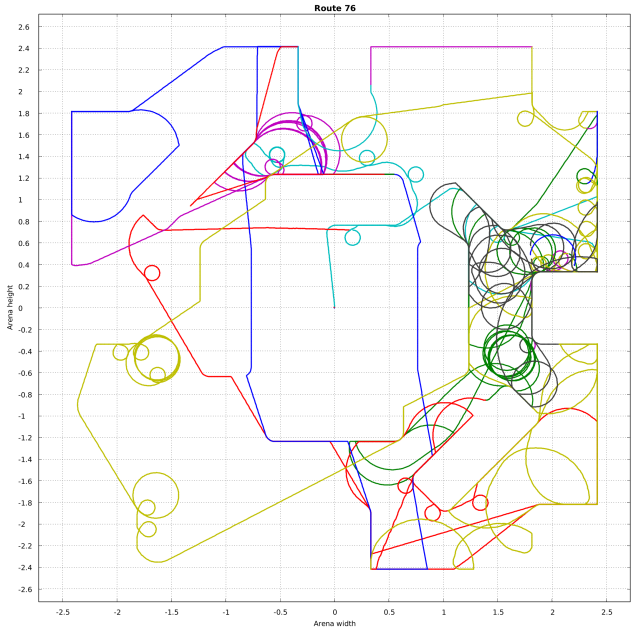


Figura 56: Tuning su arena 6.d nell'esperimento *Navigazione fra ostacoli*

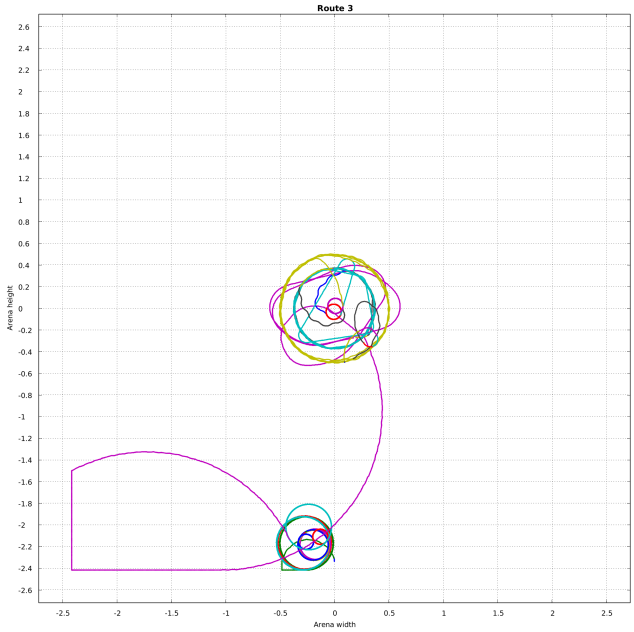


Figura 57: Benchmark su arena 6.b nell'esperimento *Ricerca spaziale assistita*

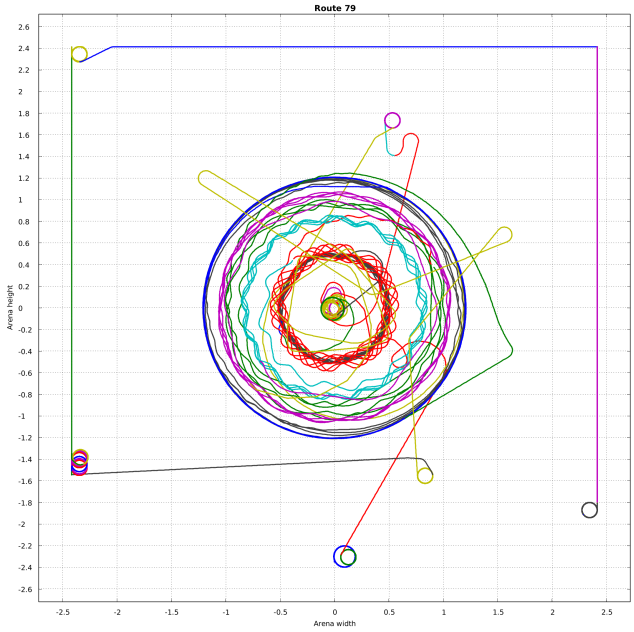


Figura 58: Tuning su arena 6.b nell'esperimento *Ricerca spaziale assistita*

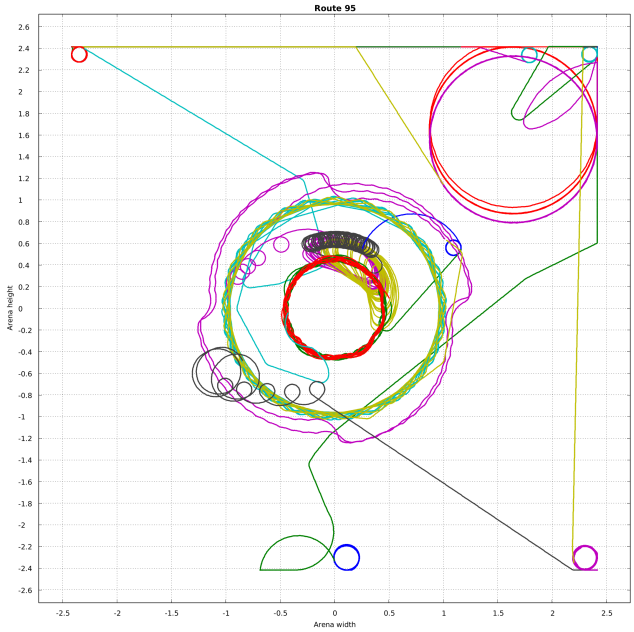


Figura 59: Benchmark su arena 6.e nell'esperimento *Ricerca spaziale assistita*

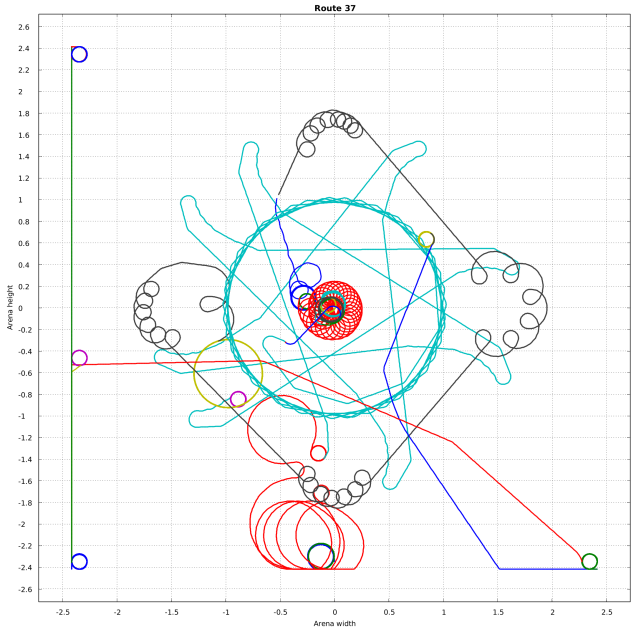


Figura 60: Tuning su arena 6.e nell'esperimento *Ricerca spaziale assistita*

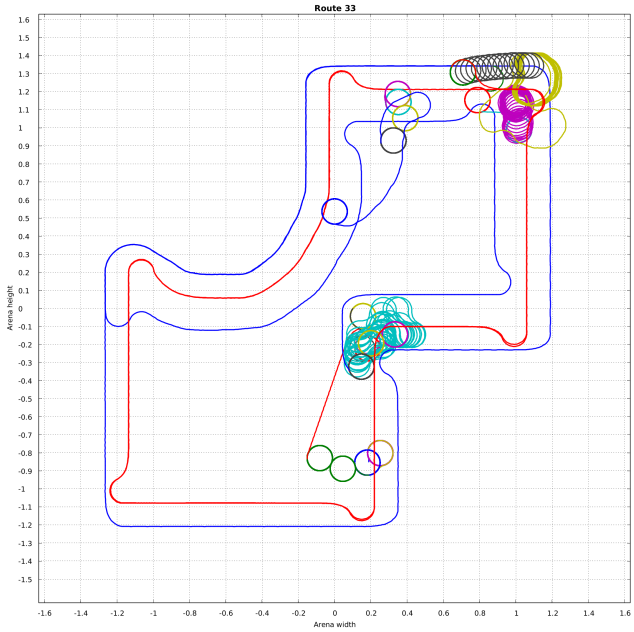


Figura 61: Benchmark su arena 6.c nell'esperimento *Navigazione su percorso*

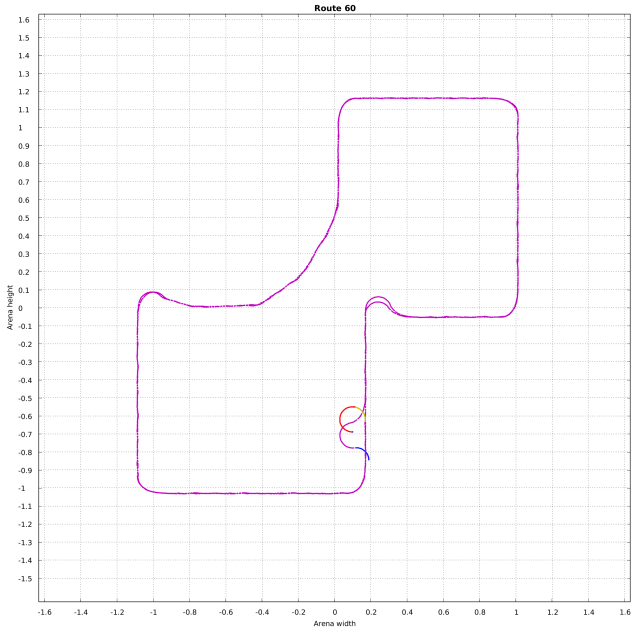


Figura 62: Tuning su arena 6.c nell'esperimento *Navigazione su percorso*

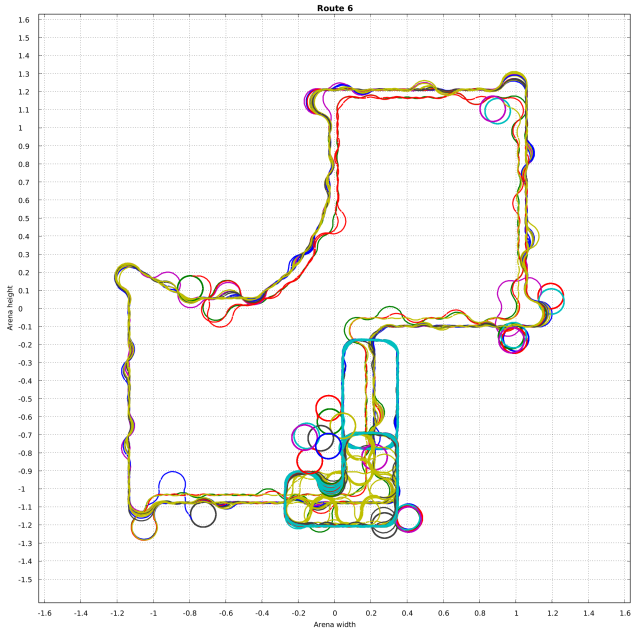


Figura 63: Benchmark su arena 6.f nell'esperimento *Navigazione su percorso*

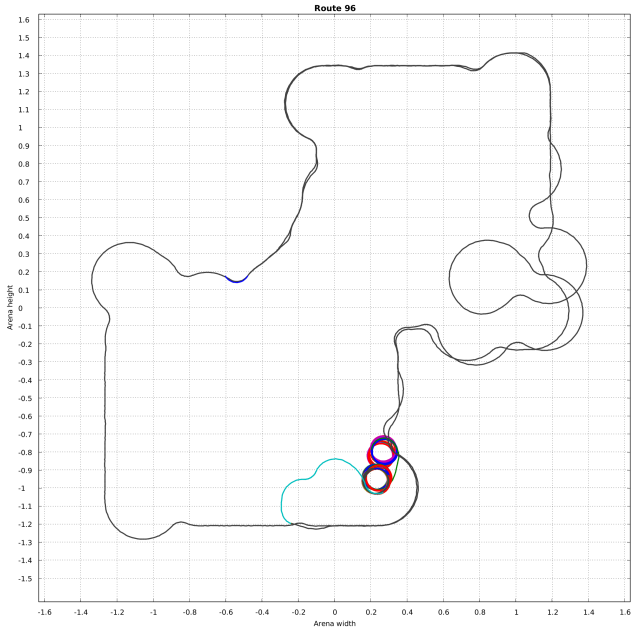


Figura 64: Tuning su arena 6.f nell'esperimento *Navigazione su percorso*