ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale in Ingegneria Informatica

EVOLUTION IN MATERIA:
esperimenti con le Nanowire Networks

Noemi Valentini

Relatore:
Prof. Dr. Andrea Roli

Correlatori:
Dr. Paolo Baldini
Dr. Michele Braccini

Anno Accademico: 2023/2024

Abstract

In questo lavoro si esplorano le possibilità dell'applicazione di tecniche di computazione evolutiva per realizzare controlli e configurazioni opportune di dispositivi fisici costituiti da Nanowire Network (NN), reti di nanofili con dinamiche simili alla plasticità sinaptica dei neuroni. Queste tecniche si pongono l'obiettivo di superare i limiti della computazione classica e avvicinarsi a forme di computazione analogica e fisica (morphological computation). Gli esperimenti condotti hanno dimostrato che le NN possono essere configurate per realizzare operazioni logiche mediante l'Evolution in Materia (EIM). L'EIM utilizza la computazione evolutiva per manipolare materiali fisici al fine di ottenere comportamenti desiderati. L'obiettivo è far evolvere prodotti fisici già esistenti piuttosto che crearne di nuovi, consentendo di trovare soluzioni innovative a problemi complessi senza conoscere a priori le caratteristiche del materiale. L'architettura delle NN è diversa per ogni nuova fabbricazione; ciò comporta che i metodi di progettazione tradizionali non sono applicabili e risulta invece cruciale l'uso di approcci evolutivi. In questo studio è stata sperimentata l'EIM con il compito di trovare le configurazioni adatte alla realizzazione di porte logiche sulle NN. Uno dei punti di attenzione di queste nuove tecnologie riguarda riproducibilità e robustezza delle soluzioni prodotte; pertanto, è stato necessario porre una particolare attenzione nell'analisi delle soluzioni trovate riguardo questi due aspetti.

Indice

Abstract **i**

1 Evolution in materia **5**

2 Nanowire Networks **11**

3 Esperimenti e implementazione **17**

3.1 Progetto 17

3.1.1 Stabilizzazione 18

3.1.2 Librerie per il genetico 19

3.2 Implementazione 21

3.2.1 Algoritmo genetico 22

3.2.2 Simulatore 24

4 Risultati e discussione **29**

4.1 Analisi a posteriori: riproducibilità e rumore 37

4.1.1 Riproducibilità e stabilizzazione 37

4.1.2 Robustezza 40

4.1.3 Migliori soluzioni per *and*, *or* e *xor* su reti configurabili 42

4.2 Discussione 46

4.2.1 Limiti dello studio e possibili miglioramenti 46

5 Conclusioni **49**

Elenco delle Figure **51**

Elenco delle Tabelle **55**

Bibliografia **55**

Ringraziamenti **61**

Introduzione

L'evoluzione naturale ha permesso che le caratteristiche favorevoli degli organismi viventi diventassero più comuni nella popolazione nel corso delle generazioni: processi di adattamento hanno portato alla realizzazione di organismi viventi sempre più complessi. I metodi tradizionali di intelligenza artificiale basati solo sulla manipolazione di simboli e regole predefinite non sono sufficienti per creare sistemi che possano eguagliare le capacità di adattamento e apprendimento tipiche degli esseri viventi. Per ottenere sistemi così elaborati ed efficienti è necessario adottare approcci che sfruttano principi e meccanismi simili a quelli presenti in natura. L'Evolution *in materia* (EIM) prende ispirazione dal mondo biologico: essa infatti manipola direttamente i sistemi fisici sfruttando la loro capacità di adattarsi all'ambiente circostante. A differenza delle classiche progettazioni convenzionali altamente vincolate, l'EIM sfrutta la computazione evolutiva al fine di manipolare direttamente materiali fisici così da ottenere i comportamenti desiderati. In questo meccanismo si osservano due domini: quello fisico e quello informatico. Nel dominio fisico, il materiale configurabile viene stimolato e analizzato, mentre nel dominio informatico un computer gestisce l'evoluzione della stimolazione del materiale tramite un algoritmo genetico. L'idea è di far evolvere prodotti fisici già esistenti piuttosto che crearne di nuovi; in questo modo sarà possibile trovare soluzioni innovative a problemi complessi senza conoscere a priori le caratteristiche del materiale. L'unica accortezza necessaria è che i materiali sottoposti a l'evoluzione dei segnali di controllo (CCE) devono essere capaci di auto-organizzarsi e adattarsi. L'EIM ha applicazioni in vari campi,

tra cui la scienza dei materiali, la robotica, l'elettronica e la nanoscienza, ed è in grado di affrontare problemi classici come lo sviluppo di porte logiche, la classificazione e il problema del commesso viaggiatore, offrendo soluzioni innovative che i metodi tradizionali non possono generare.

Il materiale su cui si è concentrato il lavoro di questa tesi sono le Nanowire Networks (NN). Questa è una tecnologia emergente introdotta nel campo del calcolo neuromorfico. Le NN sono delle reti di nanofili con caratteristiche simili a quelle di un cervello umano: comportamento memresistivo, dinamica complessa e non lineare, auto-organizzazione dei nanofili, rewiring e reweighting. A causa dell'architettura di queste reti che è unica e non può essere determinata a priori, non è possibile utilizzare metodi di progettazione tradizionali. Per questo sono stati condotti degli esperimenti in cui è stato utilizzato l'EIM per configurare le NN. Per esplorare la possibilità di questo approccio e le potenzialità di questa tecnologie si è scelto di sviluppare un compito computazionale alla base dei sistemi informatici: la realizzazione di porte logiche.

I risultati ottenuti dagli esperimenti hanno dimostrato che è possibile trovare opportune configurazioni di controllo alle NN per realizzare operazioni quali *and*, *or* e *xor*; questo è stato possibile grazie all'evoluzione artificiale del controllo adatto alla realizzazione di ogni funzione specifica. L'approccio EIM ha necessità però di tenere conto di tutta una serie di rischi in cui può incorrere: negli esperimenti si è tenuto conto delle caratteristiche della rete, della riproducibilità degli esperimenti e della robustezza delle soluzioni trovate. Si è dimostrato inoltre possibile configurare una stessa rete con più funzionalità. Purtroppo è da tenere conto che il tempo computazione ha avuto un impatto significativo nella scelta dei parametri utilizzati per gli esperimenti. L'utilizzo del simulatore delle NN, dovuto al fatto non erano disponibili le fisicamente, ha inciso notevolmente sul tempo di stimolazione della rete. La speranza è di ampliare lo studio tramite utilizzo di tecniche più ricercate come raffinare gli operatori del genetico e utilizzare di tensioni in frequenza.

Nel primo capitolo di questa tesi sarà introdotta la computazione in materia spiegando quale è il suo funzionamento e analizzando il contesto in cui è iniziata a sviluppare. Nel capitolo 2 vi sono descritte l'architettura, le caratteristiche e il processo di formazione delle Nanowire Networks. Il capitolo 3 introduce gli esperimenti delle porte logiche implementate sulle reti di nanofili auto-organizzanti tramite evoluzione di segnali di controllo: descrive le interazioni tra le componente fisica e il computer, gli operatori dell'algoritmo genetico e il processo di stimolazione sul simulatore. Nel capitolo 4 vengono raccolti e discussi i risultati ottenuti dagli esperimenti, attraverso un'analisi post evoluzione vengono verificate la qualità e la performance delle soluzioni ottenute. Infine, oltre a considerazioni conclusive, si illustreranno brevemente possibili sviluppi di questo lavoro.

Capitolo 1

Evolution in materia

L'Evolution in Materia (EIM) utilizza la computazione evolutiva per manipolare un sistema fisico; in questo modo è possibile sfruttare le caratteristiche intrinseche del sistema. L'obiettivo è di gestire un sistema fisico che tramite evoluzione di segnali di controllo (Computer Controlled Evolution, CCE), sia in grado di produrre dei comportamenti desiderati per eseguire compiti specifici. [13]

L'evoluzione naturale è una delle teorie fondamentali della biologia: individui di varie specie si modificano per adattarsi all'ambiente circostante. La necessità di sopravvivenza spinge al miglioramento delle caratteristiche genetiche, rendendole adatte all'ambiente in cui vive l'organismo. Questo concetto di adattamento è essenziale non solo in biologia, ma anche in ambiti come sistemi artificiali, psicologia, economia, controllo e intelligenza artificiale. Lo studio di queste aree, la teoria dei sistemi adattativi, è stato al centro del lavoro di Holland [8] che nel 1992 ha presentato un modello matematico in grado di spiegare tali complesse interazioni e definire gli strumenti, come gli algoritmi genetici, per manovrare tali sistemi. Gli algoritmi genetici sono una classe di algoritmi di ottimizzazione e ricerca e vengono utilizzati per trovare soluzioni ottimali a problemi complessi. Gli individui rappresentano le soluzioni al problema e vengono codificati per poter essere maneggiati dagli operatori; il loro insieme definisce una popolazione P composta da N

individui. A ogni generazione gli individui vengono valutati, ossia gli viene attribuito un punteggio di fitness che è il risultato della funzione obiettivo che determina quanto è buono quell'individuo per la risoluzione del problema. L'algoritmo genetico applica poi una selezione in base al valore che ogni individuo ha ottenuto: l'obiettivo infatti è quello di portare avanti i migliori. Gli individui selezionati subiscono una variazione, mutazione o crossover o entrambi, in modo tale da garantire la continua ricerca di nuove soluzioni migliori nelle generazioni successive. Il crossover combina le informazioni genetiche di due genitori per generare uno o più figli; la mutazione consiste nel modificare un singolo gene di un individuo ed è fondamentale per mantenere la diversità genetica all'interno della popolazione e per evitare che l'algoritmo converga prematuramente verso soluzioni subottimali. L'intero processo viene ripetuto per G generazioni o fino a quando non viene soddisfatto il criterio di arresto.

L'adattamento e il cervello diventano argomento di studio tra gli anni 50' e 60': in questo periodo infatti si svilupparono le correnti di pensiero dei cibernetici, in cui medici e psichiatri di Gran Bretagna e America concentrarono il loro lavoro attorno a questi argomenti. Il loro obiettivo era creare una macchina capace di apprendere e migliorarsi autonomamente. Sebbene i primi lavori abbiano delle differenze rispetto gli approcci attuali, hanno posto le basi per lo sviluppo dell'EIM. In questo periodo di fervore, infatti, sono emersi i primi lavori pionieristici.

Il lavoro di Pask [7] con il solfato ferroso viene riportato come tale. Il suo obiettivo era quello di creare un dispositivo sensibile al suono e ai cambiamenti magnetici, in grado di eseguire elaborazioni del segnale in modo simile a quelle di un orecchio umano. Figlio delle ricerche e degli studi di quegli anni, il progetto aveva l'obiettivo di far crescere strutture neurali tramite assemblaggi elettrochimici di struttura complessa. Per garantire questa complessità, Pask utilizzò una soluzione di metallo e sale, nella quale strutture simili ai fili metallici erano in grado di autoassemblarsi. Esse infatti modificavano la loro struttura, posizione e comportamento in risposta alla

modifica di correnti elettriche. Il risultato fu che i fili di ferro erano in grado di rispondere in modo diverso a due diverse frequenze sonore.

L'approccio di Pask definisce quindi le fondamenta per quello che sarà poi l'EIM, in quanto fu il primo a sfruttare le proprietà fisiche del solfato ferroso in modo tale che fossero in grado di auto-organizzarsi per compiere un determinato compito. Il suo lavoro può essere visto inoltre come una forma di computazione analogica. I suoi sistemi non seguono un approccio rigidamente digitale o basato su regole predefinite come i computer tradizionali, vengono bensì utilizzati componenti elettromeccanici e proprietà fisiche per rispondere e adattarsi all'ambiente. La computazione analogica utilizza interazioni fisiche per eseguire calcoli, presentando un approccio alternativo e potenzialmente più potente rispetto ai metodi di computazione classica. L'EIM si pone al centro della ricerca ancora in atto nel capire se processi fisici reali possano estendere le capacità computazionali oltre quelle del modello universale di Turing. Alcuni esempi di utilizzo di specifiche proprietà fisiche e chimiche per eseguire calcoli sono la computazione quantistica e i sistemi di reazione-diffusione [1, 5].

L'utilizzo di metodi convenzionali che impiegano solo approcci puramente simbolici non sono in grado di costruire sistemi computazionali di eguale potenza rispetto agli organismi viventi. I complessi sistemi software del futuro, se saranno in grado di sfruttare gli effetti fisici e utilizzare una forma di processo di ricerca simile all'evoluzione naturale, dovrebbero quindi produrre risultati con capacità più simili a quella dei sistemi presenti in natura. L'EIM promuove un approccio innovativo lontano da questi metodi tradizionali grazie al fatto che prende ispirazione dal mondo biologico, in cui la programmabilità non è necessaria per ottenere capacità computazionali avanzate. In particolare supera i metodi utilizzati dalla progettazione umana dove per risolvere un problema complesso si suddivide il problema in blocchi per poi combinarli tra loro in un modo altamente vincolato. Invece l'approccio evolutivo sfrutta le proprietà fisiche dal materiale e manipola direttamente il sistema fisico, non snaturandolo e rendendolo altro, permettendo di ave-

re maggiori possibilità di creare dispositivi innovativi e utili i cui principi operativi sono finora sconosciuti.[16]

Sono presenti dei livelli di evoluzione artificiale, descritti in [6], che definiscono gli spazi in cui gli operatori evolutivi operano o dove vengono prodotti i risultati.

- Il primo corrisponde al trattare tutti prodotti nel digitale senza produrre alcun artefatto fisico (evolution in simulo, ottimizzazione evolutiva).
- Il secondo livello invece progetta un artefatto fisico, o almeno qualcosa che possa essere realizzato fisicamente.
- Il terzo livello corrisponde al trattare tutti i prodotti e il processo evolutivo da cui provengono nello spazio fisico.

L'EIM corrisponde al livello 2.5 dei livelli dell'evoluzione artificiale. Il motivo per cui opera in un livello intermedio rispetto a quelli proposti è che non produce realmente un prodotto fisico, ma ne evolve uno già esistente.

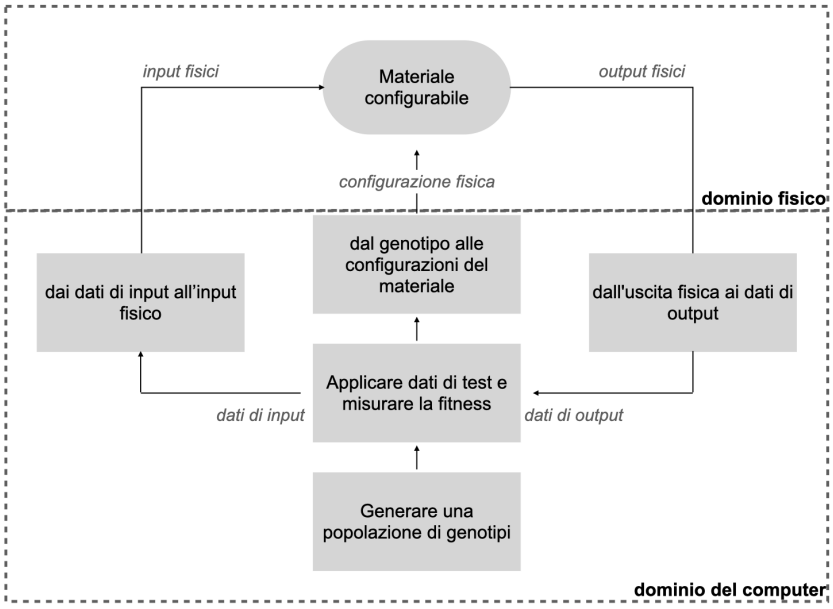

Figura 1.1: Rappresentazione schematica dell'evolution in materia, presa da [13]

Per vedere più nel dettaglio qual è il funzionamento dell'EIM è possibile osservare la figura 1.1. Innanzitutto sono presenti due domini, quello fisico e quello informatico/digitale. Nel dominio fisico è presente il materiale configurabile a cui è possibile applicare e configurare segnali fisici. Nel dominio informatico invece è presente un computer che si occuperà di gestire l'evoluzione del materiale tramite applicazione di ingressi fisici, input e configurazione, e lettura degli output. Il controllo è effettuato da un algoritmo genetico: viene generata una popolazione di possibili configurazioni del materiale che volta per volta vengono applicate al materiale. Una volta stimolato, il materiale produrrà gli output che verranno sottoposti al calcolo della funzione di fitness. Il materiale deve poter essere resettato a ogni stimolazione per far si che configurazioni passate non influenzino il calcolo della fitness successive.

I materiali fisici adatti all'EIM sono quelli che sono in grado di autoorganizzarsi e modificare le loro proprietà strutturali in risposta a cambiamenti nell'ambiente, in modo simile a come gli organismi biologici evolvono nel tempo; è importante che i materiali *imparino* a rispondere in modo ottimale a specifici stimoli.

Miller, Haring e Tufte hanno raccolto nel 2014 tutti i possibili vantaggi e svantaggi dell'utilizzo dell'EIM.[13] In particolare, oltre al fatto che questo approccio può permettere di trovare soluzioni nuove, sconosciute all'uomo, a problemi complessi, è anche utile perché non è necessario conoscere a priori il materiale configurabile che si vuole far evolvere; è possibile quindi lavorare con delle black-box dove non si vede ciò che è presente all'interno. L'evoluzione infatti è in grado di sfruttare le variabili fisiche senza conoscerle a priori. Un altro vantaggio è che, una volta conclusa l'evoluzione del sistema, il prodotto è finito e non ha necessità di essere assemblato successivamente. Infine la quantità di computazione che ha un piccolo pezzo di materiale potrebbe essere estremamente elevata.

Bisogna però avere delle accortezze su alcuni possibili rischi in cui lo sviluppo di dispositivi tramite EIM potrebbero incorrere, come quelli di tempo,

mancanza di separabilità, scarsa riproducibilità e robustezza. Questi dispositivi, inoltre, potrebbero avere gli stessi problemi della computazione analogica. Essi infatti sono fortemente specifici e non sarebbero in grado di risolvere problemi di più tipi come i computer classici. Ciononostante bisogna tenere in considerazione il tipo di applicazione interessata, in quanto i vantaggi potrebbero essere molto più adatti rispetto a quelli delle macchine di Turing. Il tempo necessario per applicare una configurazione e stimolare il materiale potrebbe essere proibitivo, ma è da tenere in considerazione che allo stesso modo è molto il tempo utilizzato per scrivere il codice per programmi per computer classici. In ogni caso sono stati analizzati i materiali proprio tenendo conto di questa possibilità: lo scopo infatti era quello di avere un insieme di materiali capaci di configurarsi in frazioni di secondo e leggere segnali di risposta velocemente. Materiali che rispondono più rapidamente ai segnali elettrici sono i più favorevoli.

L'EIM ha applicazioni in vari campi, tra cui la scienza dei materiali, la robotica, l'elettronica e la nanoscienza. È, inoltre, in grado di affrontare problemi classici come sviluppo di porte logiche, classificazione e il problema del commesso viaggiatore.[13] Infatti grazie alla manipolazione di materiali in grado di auto-organizzarsi e adattarsi, che una volta evoluti manifestano comportamenti complessi e funzionali attraverso processi simili a quelli osservati in natura, trova nuove soluzioni che i metodi tradizionali non sarebbero in grado di generare.

Capitolo 2

Nanowire Networks

In questo periodo storico, la ricerca informatica è principalmente rivolta verso tecnologie, software e hardware, che emulano il comportamento del cervello umano.

All'interno di questo studio si vuole indagare la sfera delle tecnologie fisiche in grado di sostenere la richiesta di calcolo per i modelli di intelligenza artificiale ora in sviluppo. Sembra, infatti, non essere più sufficiente la computazione classica per sostenere tale domanda. Il calcolo neuromorfico si pone l'obiettivo di creare sistemi neurali artificiali in grado di elaborare calcoli e compiti paragonabili a quelli svolti dal cervello umano. Questo tipo di ricerca vuole quindi produrre sistemi fisici con caratteristiche simili alle strutture neurali: ad esempio, la componente stocastica che, nel mondo digitale, risulta impossibile realizzare perfettamente. [12]

Le tecnologie nanometriche risultano essere molto interessanti per questo genere di ricerca: esse, infatti, possono garantire potenza di calcolo elevata per piccolissime dimensioni. Nel progetto NASCENCE viene condotto avanti lo studio introdotto nel Capitolo 1. Si tratta di far evolvere un nanomateriale - come una scatola nera - per risolvere compiti computazionali. Il nanomateriale è posizionato su una matrice di micro-elettrodi (Micro Electrode Array, MEA). La MEA è stata inizialmente progettata per poter registrare o stimolare l'attività elettrica di neuroni, cellule cardiache o altre cellule eccita-

bili. Le dimensioni sono state realizzate sufficientemente piccole così che gli elettrodi siano in grado di individuare i segnali bioelettrici. Questa miniaturizzazione ha permesso di migliorare la risoluzione spaziale e temporale delle registrazioni [15]. La MEA è stata utilizzata in questo campo perché consente di studiare, manipolare e replicare dinamiche neurali complesse, anche su tecnologie informatiche che riflettono il funzionamento del cervello. Lo stato del nanomateriale del progetto NASCENCE viene modificato tramite la MEA. È necessario, per interagire con il materiale, costruire anche un'interfaccia in grado di leggere e produrre i segnali analogici, solitamente tensioni o correnti. Si utilizza la conversione analogico/digitale perché l'evoluzione è di tipo Computer Controlled Evolution, CCE.

Tramite l'evoluzione è possibile trovare le configurazioni del materiale che, dati stimoli esterni, permettono l'esecuzione del compito computazionale desiderato. Una volta fornite le configurazioni adatte, il materiale sarà un dispositivo autonomo e riconfigurabile per il compito specifico.[4]

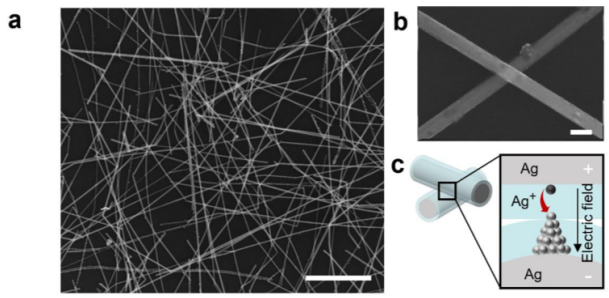

Figura 2.1: Comportamento emergente delle reti neuromorfiche auto-organizzanti a base di nanofili (NN), prese da [12]: a. Immagine al microscopio elettronico a scansione (SEM) di una rete neuromorfica auto-organizzante basata su NN altamente interconnesse (scala di 5 m) e b. dettaglio di una giunzione della NN (scala di 200 nm). c. Rappresentazione schematica del meccanismo di commutazione resistiva alle giunzioni che, sotto l'azione del campo elettrico applicato, modula la conducibilità della giunzione.

Le Nanowire Networks (NN) rappresentano una tecnologia emergente con un enorme potenziale in una vasta gamma di applicazioni tecnologiche avanzate. Tuttavia, in quanto fortemente complesse, potrebbero incorrere in problemi come mancanza di stabilità, difficoltà di fabbricazione e progettazione. Ciononostante, le NN presentano le giuste proprietà per poter riprodurre il calcolo neuromorfico ed essere sottoposte a CCE. L'utilizzo dell'EIM potrebbe permettere una risoluzione di queste problematiche e condurre alla scoperta di comportamenti ancora sconosciuti.

Le NN sono reti tridimensionali composte da fili con diametri nell'ordine dei nanometri, il cui comportamento è assimilabile a quello di un circuito elettronico. La fabbricazione avviene tramite *drop-casting*. Questo processo consiste nel far evaporare una piccolissima quantità di materiale depositato su un substrato in modo che si formino dei sottili film (strati sottili di materiale

con spessore che varia da pochi atomi a qualche micrometro). Il substrato è isolato per garantire che il flusso di corrente sia continuo, senza interferenze tra i collegamenti. La struttura di queste reti - simile a quella di un grafo - è diversa per ogni rete generata poiché la dispersione dei fili è casuale. La componente resistiva del dispositivo è data dalle giunzioni che vengono a crearsi quando questi fili si sovrappongono. Il dispositivo ha delle caratteristiche anche di memorizzazione e viene definito memresistivo: infatti, presenta una dinamica dipendente dalle stimolazioni passate. In particolare, quando una corrente elettrica viene applicata alla rete, gli ioni del metallo (ad esempio d'argento) migrano attraverso la giunzione, per creare un ponte - ossia un percorso conduttivo di argento - che collega i due nuclei metallici dei nanofili. Il filamento creato rappresenta un cambiamento significativo nella resistenza del nanofilo. La resistività infatti diminuisce e aumenta la conduttività, in quanto quest'ultima è inversamente proporzionale alla distanza dei due nanofili. Dopo un certo lasso di tempo, l'elettromigrazione degli ioni di argento smette e il sistema si trova in uno stato stabile. Una volta che la corrente elettrica non viene più applicata alla rete, gli ioni ritornano alla loro posizione iniziale così come la resistività.[3, 9, 10]

Nel 2023 diversi sono stati gli studi attorno a queste tecnologie con risultati efficaci. Zhu et al. [17] hanno studiato il modo per sfruttare la disposizione e le proprietà delle NN per creare un sistema in grado di apprendere e memorizzare sequenze di dati in modo dinamico e online. Hanno dimostrato che tali reti possono essere utilizzate per compiti di apprendimento e memoria sequenziale con alta efficienza energetica e prestazioni. Il metodo di apprendimento dinamico online che hanno implementato su questi dispositivi aveva il compito di imparare a classificare le cifre scritte a mano MNIST: il risultato ha raggiunto un'accuratezza di circa il 93.4 percento, grazie alle ricche dinamiche spaziotemporali generate dalla rete. Nello studio di Baldini et al. [3] sono state utilizzate delle NN per effettuare il controllo di robot tramite l'applicazione dell'adattamento online. Sono stati utilizzati due tipi di algoritmi: il primo applica piccole perturbazioni alla migliore soluzione cercando

di non modificare troppo le nuove configurazioni; il secondo, più *randomico*, opera cercando di non rimanere bloccato in massimi locali, a differenza del primo, e di trovare la configurazione migliore. È stato visto anche che, grazie alle proprietà delle reti di nanofili, è possibile implementare una computazione basata su Reservoir Computing (RC) alle NN. Il RC è un paradigma di machine learning sviluppato per reti neurali ricorrenti, in cui un "reservoir" (serbatoio) di neuroni o elementi ricorrenti è utilizzato per elaborare informazioni in ingresso. Il lavoro di Milano and altri [9] ha sperimentato questo approccio alle NN evolute tramite EIM, avendo come obiettivo, anche in questo caso, di risolvere un problema di classificazione di cifre scritte a mano MNIST. Attraverso questo approccio si è dimostrata la possibilità di risolvere compiti computazionali con alta efficienza energetica. Si è inoltre visto che una configurazione dove gli stessi elettrodi agiscono sia come ingressi che come uscite riduce la complessità hardware, offrendo potenziali vantaggi di costo e scalabilità rispetto a configurazioni architettonicamente simili a una matrice.

In sintesi, queste reti di nanofili - come osservato in [12] - possono adattarsi dinamicamente, simulando i processi osservati nelle reti neurali biologiche, in queste modalità:

- comportamento memresistivo, dovuto all'elettromigrazione degli ioni, che fornisce una dinamica complessa e non lineare;
- auto-organizzazione dei nanofili, che favorisce la formazione di connessione senza un controllo esterno;
- reweighting, che modifica pesi e resistenze tra le connessioni esistenti;
- rewiring, che conduce alla formazione e rimozione di connessioni tra nanofili tramite variazione delle configurazioni.

Milano et al. [12] hanno osservato anche una coesistenza di due comportamenti: il primo, deterministico, dettato dalle leggi fisiche e dalle interazioni prevedibili nel sistema, il secondo, stocastico, derivato dal rumore e dalle

cadute di conduttanza che si verificano quando il sistema è in uno stato stazionario. Questa ricerca suggerisce che, comprendendo e sfruttando sia gli aspetti deterministici che quelli stocastici delle dinamiche della rete, si potrebbe creare un hardware che elabora le informazioni in modo più efficiente e versatile, come il cervello umano che sfrutta sia processi prevedibili che casuali.

In conclusione, le proprietà delle NN suggeriscono la possibilità di avere un forte impatto nel mondo tecnologico che le rende un ottimo ambito di ricerca e sviluppo per il calcolo neuromorfico. Il loro comportamento memresistivo permette di superare i costi legati al trasferimento di informazioni e diminuire il consumo energetico.Inoltre, essendo possibile riprodurre fenomeni analoghi a quelli della plasticità sinaptica, questi dispositivi permetteranno di sviluppare sistemi di intelligenza artificiale efficienti.

Capitolo 3

Esperimenti e implementazione

Come è stato visto nei capitoli precedenti le NN sono una tecnologia emergente nello scenario del calcolo neuromorfico; per tale ragione, è cruciale approfondire le loro capacità e applicazioni. I classici metodi di progettazione hanno difficoltà a maneggiare la loro l'architettura perché la progettazione di ogni NN è a priori sconosciuta e unica. È necessario quindi utilizzare approcci più innovativi, come quello descritto nel Capitolo 1, i quali permettono di non perdere delle caratteristiche chiave del materiale. Le NN manifestano tutte le caratteristiche necessarie per essere sottoposte a CCE: sono in grado di auto-organizzarsi e configurarsi in tempi veloci.

In questa tesi, il caso di studio è l'implementazione di porte logiche sulle NN tramite l'EIM. Per gli esperimenti non sono state utilizzate delle NN fisiche, bensì le reti sono state generate tramite l'utilizzo del simulatore [2]. L'algoritmo genetico è stato implementato tramite la libreria DEAP [14].

3.1 Progetto

Le porte logiche, per cui sono state ricercate le configurazioni delle NN, sono *and*, *or* e *xor*, in quanto ritenute essere le più significative.

Gli esperimenti sono stati condotti tenendo conto di possibili problematiche che le NN avrebbero potuto produrre: la scarsa riproducibilità è stata

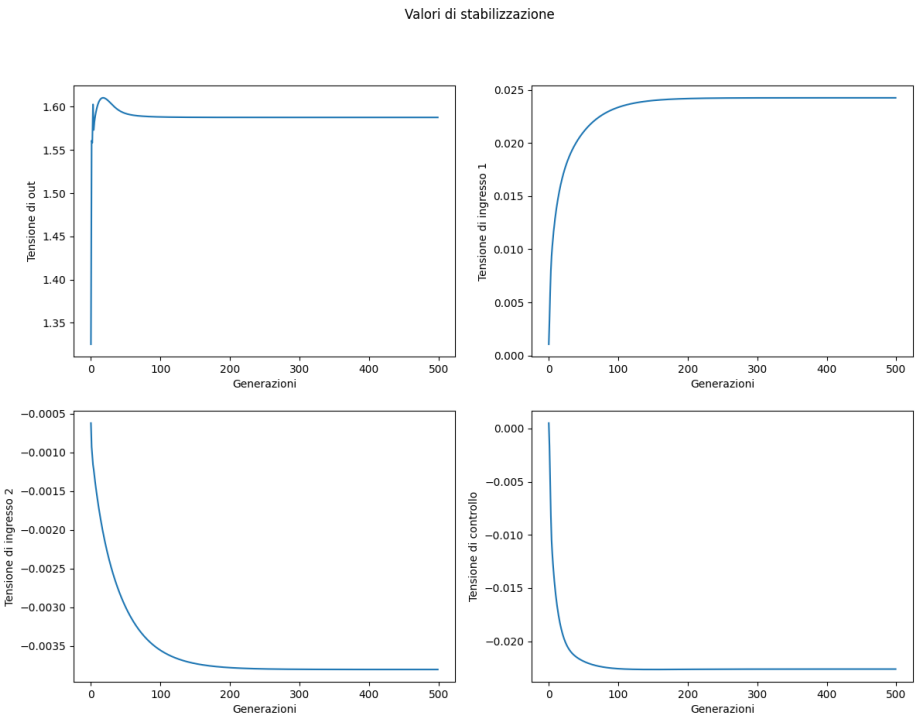

Figura 3.1: Passi di stabilizzazione sotto stimolazione: output, due ingressi e controllo.

affrontata tramite l'analisi a posteriori delle soluzioni trovate mentre la robustezza è stata affrontata tramite l'analisi a posteriori e l'introduzione del rumore durante il genetico. Questi ultimi aspetti verranno affrontati più nel dettaglio nel Capitolo 4.

3.1.1 Stabilizzazione

Come descritto nel capitolo 2, la rete, una volta stimolata, presenta una dinamica nel tempo fino a che non raggiunge un nuovo stato di *equilibrio*. Bisogna dunque tener conto che, una volta applicata una tensione, sarà necessario attendere un tempo di stabilizzazione. Il numero di passi utili alla stabilizzazione della rete deve essere scelto in modo tale che sia sufficien-

temente grande per permettere ai risultati di essere replicabili nelle analisi a posteriori, ma non così elevato da avere un impatto eccessivo sul tempo computazionale.

La figura 3.1 descrive il comportamento della rete sotto stimolazione di 3 tensioni di ingresso - 5V, 2V e 5V - applicate rispettivamente ai pin identificativi degli elettrodi 3, 6, 12; il pin 15 è collegato a massa mentre l'output viene letto dal pin 10. Questo tipo di assetto è stato scelto per iniziare a effettuare delle osservazioni della rete con una struttura simile a quella utilizzata negli esperimenti. Dalla figura è possibile osservare che sono necessari 300 passi affinché i valori siano tutti stabilizzati; negli esperimenti, tuttavia, bisogna tenere in considerazione che la durata della stimolazione non sia eccessiva. Il tempo computazionale è un altro aspetto molto significativo: la quantità di tempo dedicata alla stimolazione della rete è quella che ha l'impatto più significativo sul tempo complessivo. Dunque, è necessario trovare un punto d'incontro tra riproducibilità e tempo computazionale.

3.1.2 Librerie per il genetico

Si è scelto di utilizzare una libreria in Python per implementare l'algoritmo genetico, piuttosto che scrivere il codice *ex novo*. Di seguito, una panoramica di tre possibili librerie.

- PyEvolve: offre una serie di algoritmi evolutivi, tra cui algoritmi genetici e programmazione genetica. È facile da usare e supporta l'implementazione di algoritmi evolutivi paralleli. Purtroppo non è più attivamente sviluppata e potrebbe mancare di alcune delle ultime funzionalità o correzioni di bug.

- GAFT (Genetic Algorithm Framework): è leggera e potente per la progettazione e l'implementazione di algoritmi genetici. La sua struttura la rende facile da imparare e usare, senza però perdere in efficienza e velocità nell'esecuzione degli algoritmi. Il problema è che a causa della sua semplicità alcune funzionalità avanzate, presenti in librerie

più complesse (come DEAP), potrebbero essere assenti. Infine, la comunità di sviluppatori potrebbe non essere così grande come per altre librerie più popolari, di conseguenza potrebbe essere più difficile trovare soluzioni a problemi specifici.

- DEAP (Distributed Evolutionary Algorithms): è la libreria più popolare per implementare algoritmi genetici in Python. Possiede una vasta gamma di strumenti per la creazione di algoritmi evolutivi complessi. È altamente personalizzabile e offre molte opzioni per definire operatori genetici personalizzati. Supporta l'implementazione di algoritmi evolutivi distribuiti. Grazie alla sua notorietà, è ben documentata e con una comunità attiva di sviluppatori. Il suo unico contro è che, a volte, l'API può risultare complessa.

Per motivi di completezza, efficienza, attualità e strumenti offerti si è scelto di utilizzare DEAP.

3.2 Implementazione

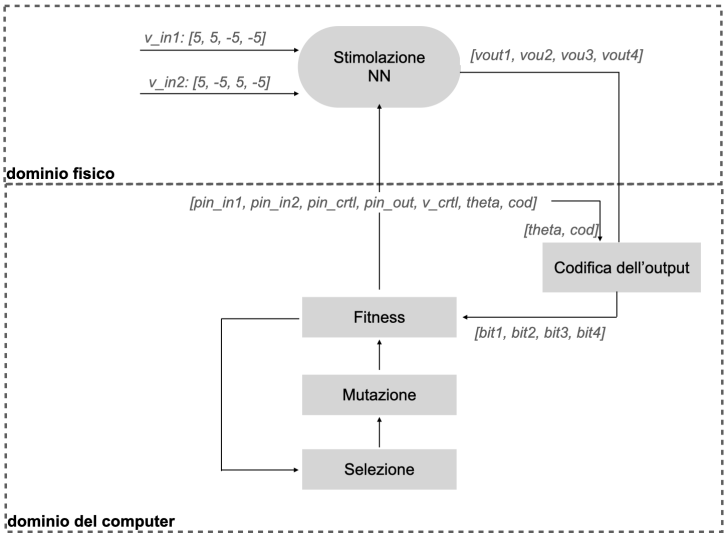

Figura 3.2: Rappresentazione schematica dell'evolution in materia applicata alle NN per la realizzazione delle porte logiche

Nella figura 3.2 è mostrato il procedimento dell'evoluzione delle NN utilizzato negli esperimenti. Innanzitutto viene creata una popolazione di individui, che rappresentano le configurazioni; ogni individuo viene sottoposto alla funzione di fitness, la valutazione consiste nello stimolare il materiale secondo la configurazione. Durante la stimolazione, viene applicata al materiale una successione di tensioni in ordine, al fine di garantire l'esplorazione di tutte le possibili combinazioni da sottoporre alla porta logica. Ogni volta che vengono applicate le tensioni di ingresso viene applicata anche la tensione di controllo. Sarà l'algoritmo genetico a indicare quali saranno gli elettrodi a cui applicare le tensioni e leggere l'output. A ogni applicazione è necessario attendere un tempo di stabilizzazione, si veda la sezione 3.1.1, prima di leggere gli output. Una volta letti i segnali elettrici avviene la codifica dei medesimi in valori binari, seguendo le direttive di soglia e tipo di codifica prodotte dall'algoritmo genetico. Successivamente, gli output codificati possono essere valutati dalla funzione di fitness. Queste sono principalmente le

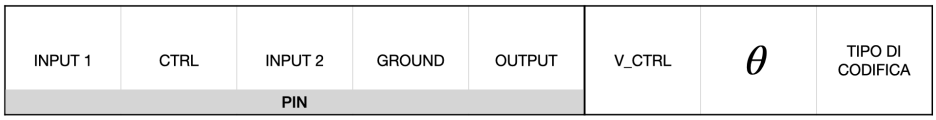

Figura 3.3: Descrizione dettagliata degli individui

interazioni del genetico con il materiale, i rapporti tra il dominio fisico e il dominio digitale del nostro sistema. Si procede, dunque, a descrivere più nel dettaglio le dinamiche del genetico.

3.2.1 Algoritmo genetico

L'algoritmo genetico utilizzato per far evolvere il materiale risulta essere semplice. Dato che gli esperimenti si inseriscono in una fase di esplorazione iniziale delle capacità delle NN, si è optato per cercare di comprendere come riuscire a ottenere risultati con metodi più essenziali, privi di elementi superflui. **Codifica degli individui**: i primi esperimenti hanno visto una codifica più semplice di quella utilizzata in figura 3.2. L'individuo era composto soltanto dal pin (identificativo dell'elettrodo ammissibile) a cui applicare tensione di controllo, dalla tensione di controllo e dalla soglia per la codifica degli output. Purtroppo, con questa rappresentazione il genetico non riusciva a catturare tutti gli aspetti essenziali perché la rete realizzasse correttamente il comportamento desiderato. Dunque, la codifica finale degli individui, vedi la figura 3.3, è stata rappresentata come segue: oltre il pin di controllo vengono sottoposti al controllo genetico anche i pin di input, output, ground, e oltre il θ, per definire la codifica viene inserito un binario che indica se la codifica debba essere effettuata in maniera diretta o indiretta. In questo modo si è potuto eliminare l'aspetto altamente vincolante dei pin di input e output fissi e tenere in considerazione la possibilità di una codifica diversa a seconda del compito da realizzare.

Il genetico utilizza gli operatori descritti sotto.

- **Selezione**: selezione di tipo proporzionale, dove la probabilità di sele-

zione di un individuo è proporzionale alla sua qualità o fitness.

- **Sopravvivenza**: la nuova popolazione viene creata mantenendo i migliori K della vecchia popolazione e generando gli altri con la mutazione.

- **Mutazione**: si verifica per ogni gene dell'individuo ma solo se viene superata una certa probabilità di mutazione pm. A seconda del tipo di gene avverrà una modifica diversa: i pin cambiano il valore casualmente, ne viene scelto un altro tra quelli ammissibile (non si vuole che il pin venga utilizzato più volte nello stesso individuo); alla tensione di controllo si somma algebricamente un valore estratto da una distribuzione gaussiana di media (μ) 0 e deviazione standard (σ) 0.1. Se il valore risulta essere fuori dall'intervallo [Vmin, Vmax] viene troncato. La soglia (θ) segue la stessa modifica della tensione di controllo. Il binario che descrive il tipo di codifica da effettuare viene invertito.

- **Fitness**: si stato scelto di utilizzare una frazione di successi sulle 4 possibili combinazioni di ingressi. Più formalmente:

$$F(i) = \frac{1}{4} \sum_{i=1}^{4} f_i \tag{3.1}$$

dove

$$f_i = \begin{cases} 1 & \text{se l'uscita è corretta} \\ 0 & \text{altrimenti} \end{cases}$$

Si è considerato un esperimento per ogni funzione logica: ad esempio, dati i segnali di ingresso pari a [Vmin, Vmax, Vmin, Vmax] e [Vmin, Vmin, Vmax, Vmax], verrà verificato quanto sarà simile l'output codificato rispetto la successione di output desiderati, che nello specifico sono: [0, 0, 0, 1] per l'*and*, [0, 1, 1, 1] per l'*or* e [0, 1, 1, 0] per lo *xor*.

Iperparametri: l'algoritmo genetico è stato fatto evolvere per 100 generazioni come descritto in tabella 3.1. A ogni generazione è stata effettuata

Generazioni	100
Popolazione	50
K-top	10
Probabilità di mutazione	0.2

Tabella 3.1: Iperparametri del genetico

una verifica: l'evoluzione termina qualora il valore di fitness di un individuo raggiunga il massimo possibile. La scelta di probabilità di mutazione invece è dipesa dalla necessità di non volere variazioni troppo grandi tra le configurazioni trovate; rimanendo il più possibile vicino alle soluzioni ottenute a ogni generazione. Per lo stesso motivo, si è scelto di non utilizzare il crossover. Durante gli esperimenti inizialmente era stata utilizzata una probabilità di mutazione pari a 0.125 ($1/n$ dove n è il numero di geni dell'individuo), ma si è visto che risultava essere molto difficile superare questa probabilità e che durante l'evoluzione si avevano molte valutazioni di individui uguali. Per questo si è portata la probabilità di mutazione a 0.2.

3.2.2 Simulatore

Gli esperimenti sono stati svolti sul simulatore fornito da Baldini [2]. Il simulatore proposto, è basato sul modello di Milano et al. [10]. Baldini [2] cercando di renderlo più agile per il controllo dei robot, ha aggiunto la possibilità di definire i pin, identificativi degli elettrodi, come *source*, *load* o *ground*, permettendo così di includere sensori e motori. È stato utilizzato un simulatore perché la rete di nanofili non era disponibile fisicamente. Questa scelta ha permesso di fare un'analisi diretta senza passare per tutta una seria di conversioni di elementi fisici e di elementi reale. Inoltre è stato anche possibile effettuare più facilmente dei test di diverse configurazioni, così da avere una raccolta di dati statistici più vasta. D'altra parte, però, l'utilizzo del simulatore richiede molto più tempo di calcolo, rendendo necessaria una limitazione sulla durata delle simulazioni.

Numero di fili	200
Lunghezza media dei fili	40.0
Deviazione standard della lunghezza	14.0
Dimensione del pacchetto	500
Seme di generazione	[1...10]

Tabella 3.2: Scheda tecnica delle NN

Si discuterà ora un po' più nel dettaglio quali sono gli aspetti delle NN necessari per comprendere il funzionamento del simulatore. Le connessioni definiscono il comportamento resistivo delle NN e dipendono dal numero di fili, dalla lunghezza e dalla dimensione del pacchetto del dispositivo. Per descrivere come è composta la struttura delle NN, è sufficiente utilizzare una misura derivata di questi parametri: la densità normalizzata. Questa è calcolata secondo la seguente formula:

$$D = N\frac{L^2}{S^2} \tag{3.2}$$

Dove:

- D è la densità normalizzata;
- N è il numero di nanofili;
- L è la lunghezza media dei fili;
- S è la dimensione di un lato del pacchetto quadrato.

La densità normalizzata determina la distribuzione delle componenti connesse. I parametri delle reti, utilizzati per gli esperimenti, sono quelli descritti in tabella 3.2.

La scheda tecnica descrive le caratteristiche statiche della rete necessarie per generarla. Il seme di generazione garantisce la riproducibilità della rete sul dispositivo dove si sta lavorando. Nella tabella è definito come un range che va da 1 a 10 perché per gli esperimenti sono state generate 10 reti diverse al variare di questo valore.

Di seguito è stata riportata una porzione di codice in cui viene simulata la stimolazione della rete.

```
{c}
 def stimulation(self, individual):
        pins=individual[0]
        v=individual[1]

        #Selezione dei pin come I/O/C
        OUTPUT = pins[4]
        #SOURCE: i1
        self.mea.ct[pins[0]] = 1
        #SOURCE: ctrl
        self.mea.ct[pins[1]] = 1
        #SOURCE: i2
        self.mea.ct[pins[2]] = 1
        #GROUND
        self.mea.ct[pins[3]] = 2

        # Conversione MEA in interfaccia
        it = nns.mea2interface(self.mea)

        Vs = []
        #Definizione ordine delle tensioni in ingresso
        config, cv = utils.input_conf(pins[:3], v, self.noise)
        for c in config:
            # Tempo di rilassamento
            for _ in range(200):
                ios = (c_double * 3)(0, 0, 0)
                nns.update_conductance(self.ns, self.cc)
                nns.voltage_stimulation(self.ns, self.cc, it, ios)
```

```
        # Stimolazione degli Input
        for _ in range(200):
            ios = (c_double * 3)(*c)
            nns.update_conductance(self.ns, self.cc)
            nns.voltage_stimulation(self.ns, self.cc, it, ios)

        # Letura output
        for _ in range(2):
            ios = (c_double * 3)(*cv)
            nns.update_conductance(self.ns, self.cc)
            nns.voltage_stimulation(self.ns, self.cc, it, ios)
            output= self.ns.Vs[self.mea.e2n[OUTPUT]]
            Vs.append(output)

    #Reset dei pin
    self.mea.ct[pins[0]] = 0
    self.mea.ct[CTRL] = 0
    self.mea.ct[pins[2]] = 0
    self.mea.ct[pins[3]] = 0
    return Vs
```

Come prima operazione del metodo, si ha l'assegnazione del tipo di pin seguendo le indicazioni descritte dall'individuo prodotto dal genetico. Dopo aver configurato la MEA, che nel simulatore è composta da 16 elettrodi, viene creata l'interfaccia necessaria per simulare le connessioni tra la NN e l'esterno. Vengono definite le successioni di tensioni da applicare al simulatore: le successioni delle tensioni di ingresso sono sempre le stesse e hanno tensione massima a 5V e tensione minima -5V. Inizialmente le tensioni erano tra [0, 5V] ma in questo modo non si predisponeva la rete a considerare la differenza tra 0 logico e 0 V. All'interno del primo *for* è presente la fase di stimolazione, la quale è a sua volta divisa in 3 momenti: si inizia con fase di rilassamento, vengono applicate tensioni 0V a tutti gli ingressi per riportare

il sistema a uno *stato iniziale*, poi vengono applicate le tensioni in ingresso, infine si ha una fase di lettura degli output. Questa stimolazione viene effettuata ogni volta che viene applicata una coppia di tensioni in ingresso. Infine vengono resettati i pin della MEA, identificandoli tutti come *none*, affinché non vengano influenzate le operazioni successive.

Capitolo 4

Risultati e discussione

L'obiettivo degli esperimenti condotti è stato quello di trovare le configurazioni delle NN tramite un algoritmo genetico per implementare le porte logiche *and*, *or* e *xor*. Dato che ogni fabbricazione genera una NN diversa dalle altre, per ottenere dei dati sufficienti a descrivere una prospettiva generale delle dinamiche di questa tecnologia, ogni esperimento è stato fatto evolvere per 10 repliche. Durante il lavoro l'interesse è stato principalmente diretto verso le analisi dei risultati della porta *xor*, la cui realizzazione comporta delle implicazioni più complesse rispetto le altre due porte: essa infatti, non essendo separabile, è in grado di descrivere meglio le caratteristiche di non linearità della rete neuromorfica. L'andamento della funzione di fitness durante le generazioni per ognuna delle 10 repliche è visibile nei grafici sotto descritti, in particolare dalla figura 4.1 alla figura 4.5 per lo xor, dalla figura 4.6 alla figura 4.10 per l'*and*, e dalla figura 4.11 alla figura 4.15 per l'*or*. In prima analisi, si osserva che è presente almeno un caso per tutte le porte in cui si trova l'individuo ottimo alla prima generazione. Nello specifico l'algoritmo genetico della porta *and* riesce a trovare 2 ottimi su 500 individui, 3 su 500 nel caso della porta *or*, 1 su 500 nel caso della porta *xor*. Dai grafici è possibile anche osservare che solitamente, qualora non si riesca a trovare soluzione ottima, la media della fitness rimane intorno al 75.

L'andamento mostra anche che la porta *or* è quella che trova più facil-

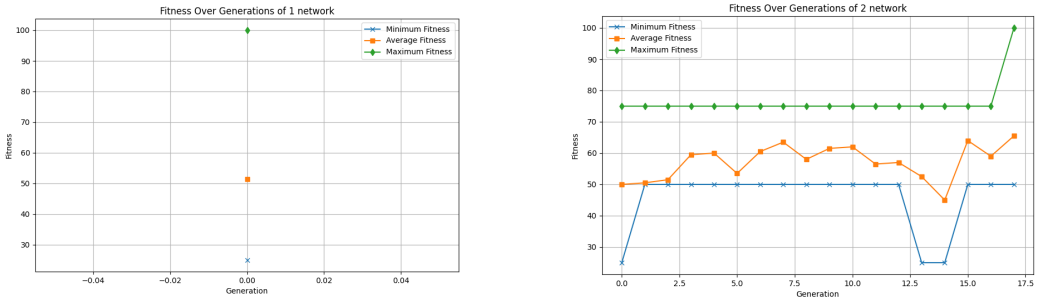

Figura 4.1: Andamento della fitness della porta *xor* delle reti 1 e 2

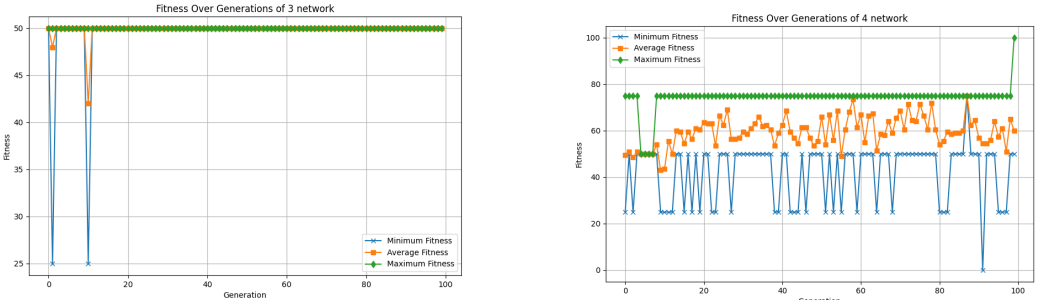

Figura 4.2: Andamento della fitness della porta *xor* delle reti 3 e 4

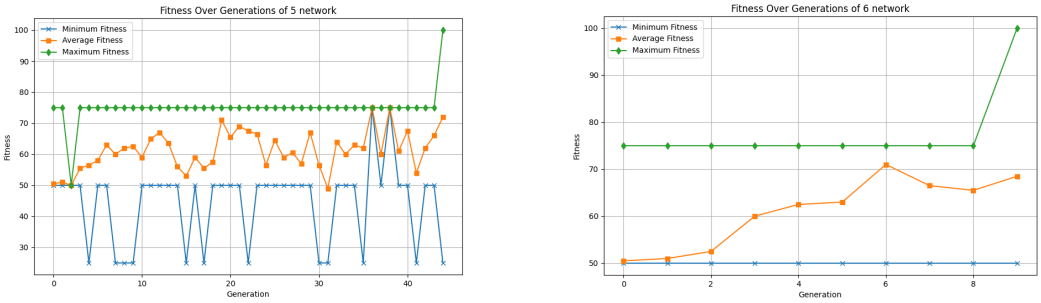

Figura 4.3: Andamento della fitness della porta *xor* delle reti 5 e 6

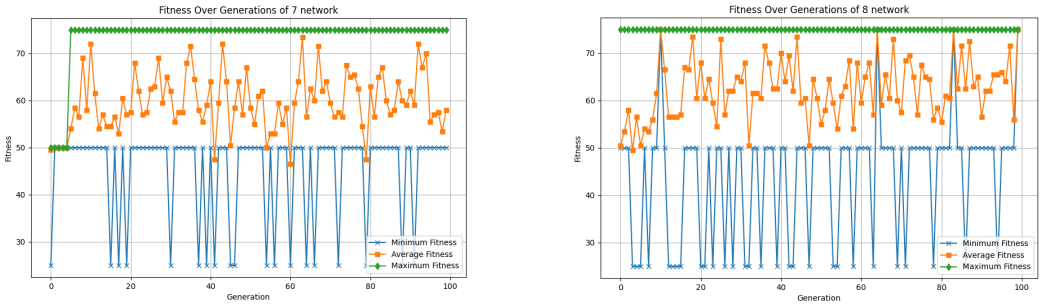

Figura 4.4: Andamento della fitness della porta *xor* delle reti 7 e 8

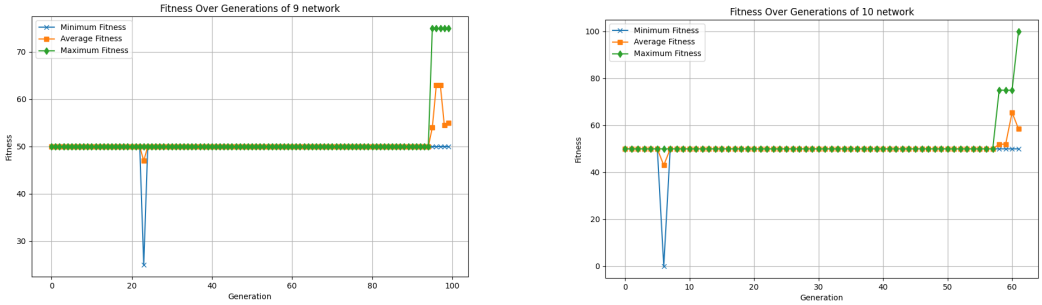

Figura 4.5: Andamento della fitness della porta *xor* delle reti 9 e 10

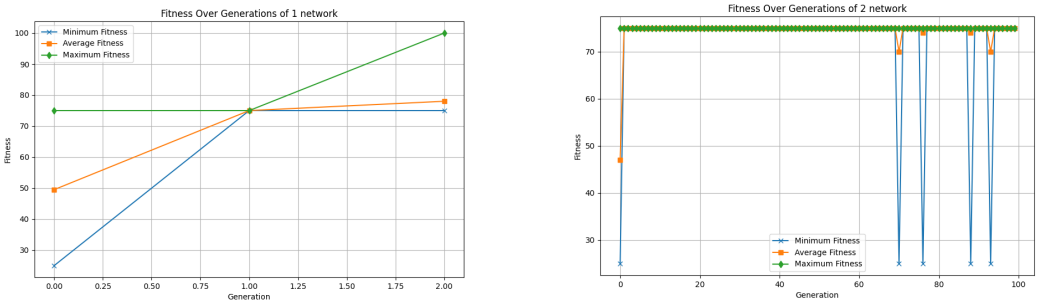

Figura 4.6: Andamento della fitness della porta *and* delle reti 1 e 2

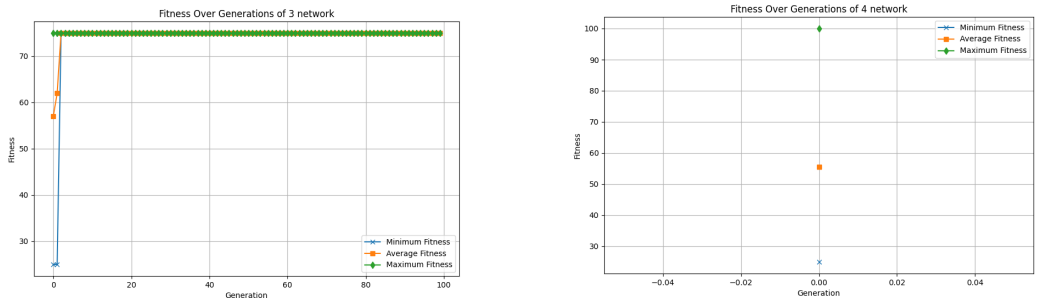

Figura 4.7: Andamento della fitness della porta *and* delle reti 3 e 4

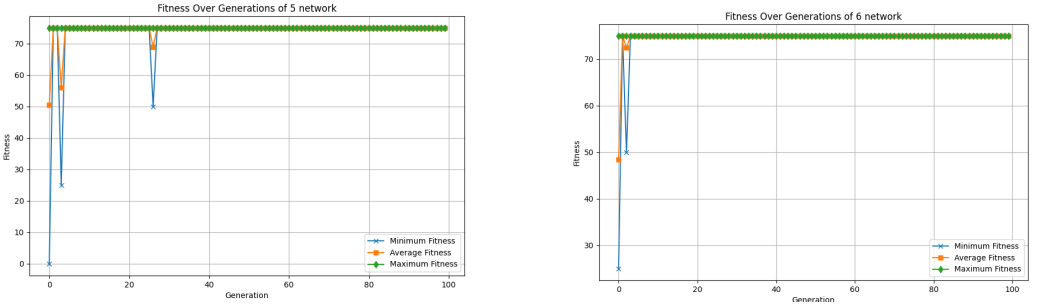

Figura 4.8: Andamento della fitness della porta *and* delle reti 5 e 6

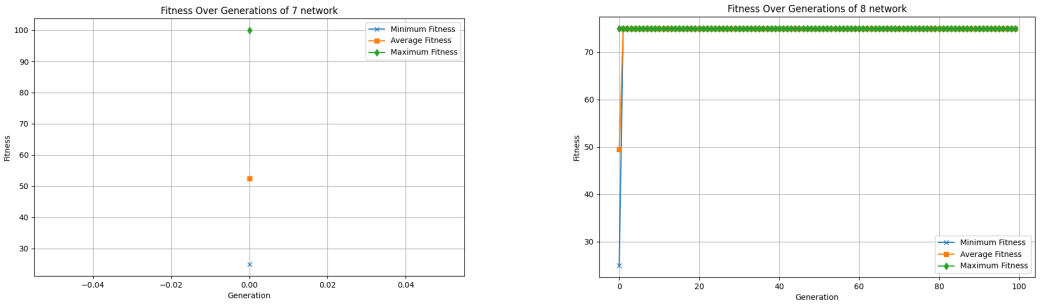

Figura 4.9: Andamento della fitness della porta *and* delle reti 7 e 8

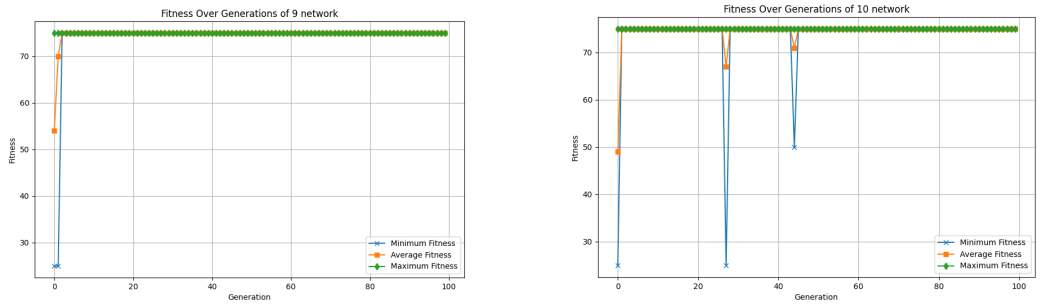

Figura 4.10: Andamento della fitness della porta *and* delle reti 9 e 10

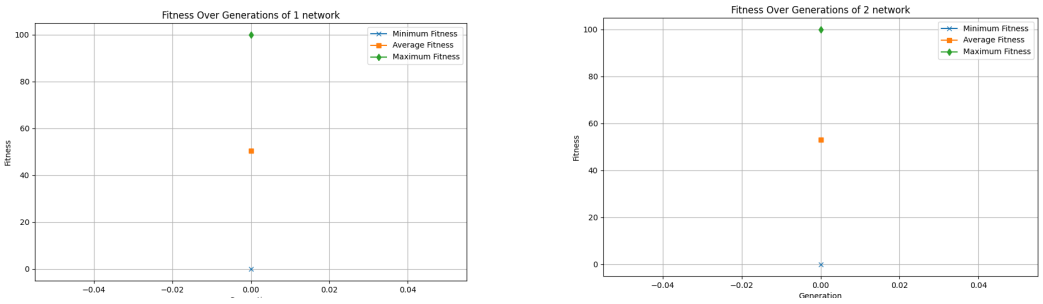

Figura 4.11: Andamento della fitness della porta *or* delle reti 1 e 2

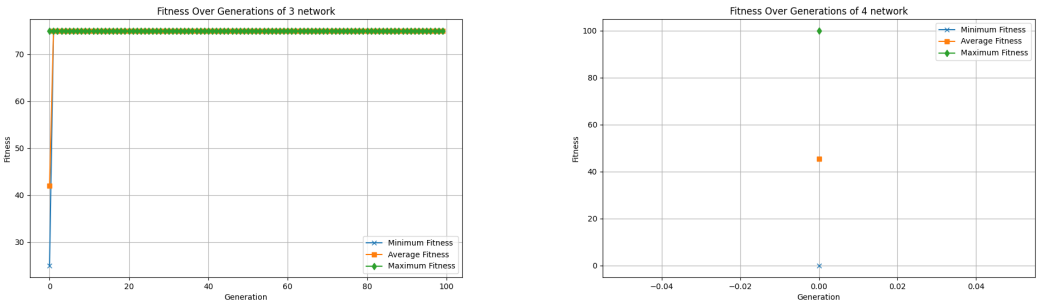

Figura 4.12: Andamento della fitness della porta *or* delle reti 3 e 4

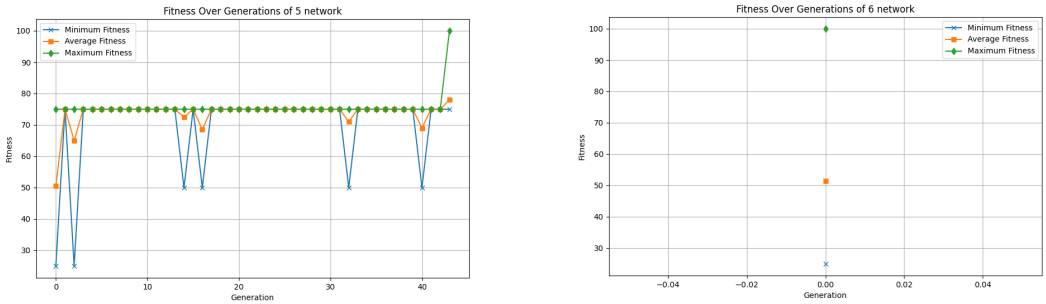

Figura 4.13: Andamento della fitness della porta *or* delle reti 5 e 6

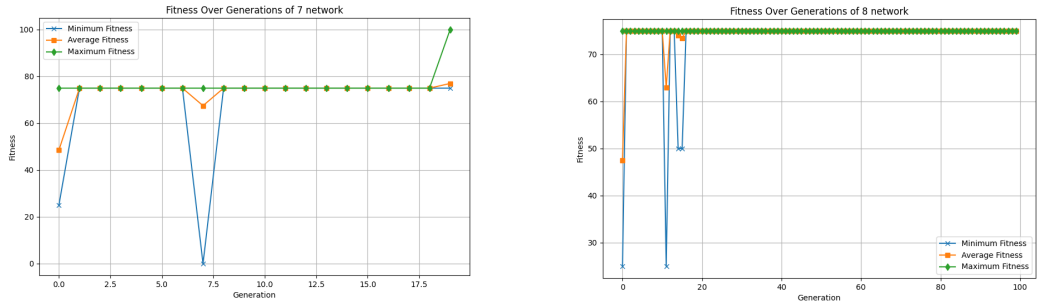

Figura 4.14: Andamento della fitness della porta *or* delle reti 7 e 8

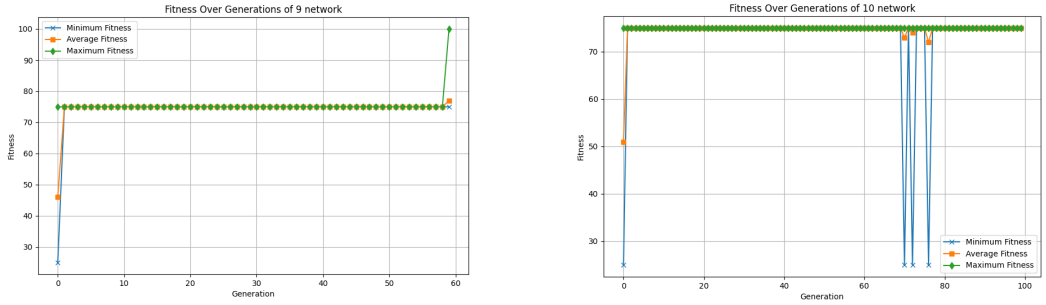

Figura 4.15: Andamento della fitness della porta *or* delle reti 9 e 10

	AND	OR	XOR
Genetico su 10 reti	3	7	6
Altre reti	13	24	15
Generazione 0 su 500 individui	2	3	1

Tabella 4.1: Risultati dell'algoritmo genetico

mente soluzioni; invece risulta essere l'*and* quella che ottiene meno risultati, al contrario delle aspettative ottenute durante i primi esperimenti in cui la porta *xor* sembrava essere la più ostica. In ogni caso, dal momento che le due porte *and* e *or* sono una l'inversa dell'altra, è sufficiente che una delle due ottenga buoni risultati per descrivere opportunamente le potenzialità della tecnologia NN.

È interessante, inoltre, notare che gli individui migliori per le porte *or* e *and* sono tutti in codifica indiretta, mentre solo due individui della porta *xor* risultano codificare gli output in modo convenzionale.

Le soluzioni prodotte dall'algoritmo genetico per ogni replica sono riassunte nella tabella 4.1.

Nel corso di questi esperimenti è stato anche osservato che le reti 2, 7 e 10 hanno un elettrodo che non è connesso alla rete nello stesso modo degli altri: nello specifico, per la rete 2 e 7 non è connesso il pin 1, per la rete 10 il pin 9. I pin non connessi non vengono mai utilizzati come soluzione dell'evoluzione; la causa deriva dal fatto che, qualora uno di questi pin venisse utilizzato, non porterebbe alcun contributo alla configurazione del circuito e mancherebbe una componente necessaria alla realizzazione della porta logica. Nello specifico si è notato che, se l'elettrodo non connesso corrisponde o al pin da cui viene applicata la tensione di controllo o a quello da cui viene effettuata la lettura, l'output è sempre 0V.

Nelle figure 4.16 si osservano le soluzioni trovate per la rete generata dal seme 1. Il grafico mostra la struttura a grafo della rete, introdotta nell'articolo [11]: ogni nodo rappresenta un nanofilo mentre ogni arco una giunzione. I colori dei nodi mostrano le tensioni sul nanofilo e i colori degli archi rap-

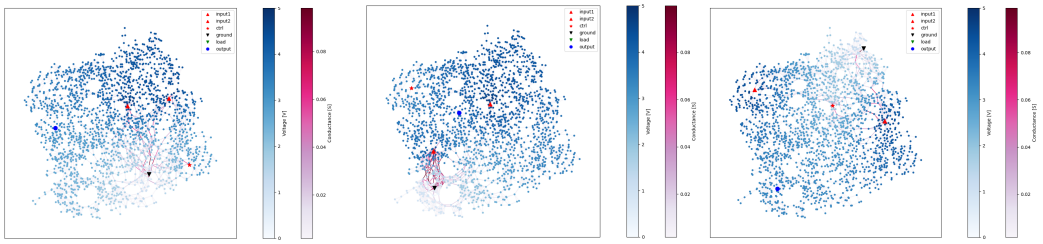

Figura 4.16: Visualizzazione grafica della stimolazione della rete 1 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 10 e 3, di ctrl 1, di ground 4 e output 6, tensione di controllo circa 3.1V, θ circa 0.7; xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5

presentano la loro conduttività, entrambi nel momento in cui è stato memorizzato lo stato della rete. La legenda mostra i nodi di *input*, *ctrl*, *ground* e *output*. Prendendo in esempio la figura 4.16, il comportamento è dato dalle configurazioni per le rispettive porte sotto indicate.

- *And*: i pin 10 e 14 per le tensioni di ingresso, il pin 16 per la tensione di controllo, intorno a 3.01V, il pin 12 viene collegato a massa e il pin 2 è stato utilizzato per la lettura dell'output. La codifica di tipo indiretta ha tradotto la tensione di output di 0.1 come binario 1, essendo $\theta = 0.2$.

- *Or*: i pin 10 e 3 per le tensioni di ingresso, il pin 1 per la tensione di controllo, intorno a 3.1V, il pin 4 viene collegato a massa e il pin 6 è stato utilizzato per la lettura dell'output. La codifica di tipo indiretta ha tradotto la tensione di output di 0.50 come binario 1, essendo $\theta = 0.7$.

- *Xor*: i pin 15 e 1 per le tensioni di ingresso, il pin 10 per la tensione di controllo, intorno a 1.3V, il pin 13 viene collegato a massa e il pin 4 è stato utilizzato per la lettura dell'output. La codifica di tipo indiretta ha tradotto la tensione di output di 0.55 come binario 0, essendo $\theta = 0.50$.

4.1 Analisi a posteriori: riproducibilità e rumore

4.1.1 Riproducibilità e stabilizzazione

All'inizio di questi esperimenti, le soluzioni della porta *xor* erano state sottoposte a un analisi a post evoluzione. Le configurazioni delle reti, che avevano portato il termine dell'algoritmo genetico, sono state riproposte alla rete per verificare la loro capacità di realizzare il compito computazionale richiesto. Purtroppo però, gli output restituiti durante questa seconda analisi, sono risultati diversi e non più in grado di eseguire il comportamento desiderato. Questa revisione ha evidenziato l'eventualità di un problema di riproducibilità degli esperimenti. Per risolvere questo primo ostacolo è stato necessario aumentare i passi di stabilizzazione della rete - da 100 a 200 - avendo così una tolleranza più alta, per cui i valori risultano essere uguali. La scelta di 200, come osservato nella sotto sezione 3.1.1, tiene conto sia del tempo computazionale che della riproducibilità della porta logica ottenuta.

Una volta utilizzato un numero maggiore per stabilizzare la rete, i risultati degli output si sono mostrati simili nella verifica a posteriori rispetto agli output ottenuti durante il genetico, quindi è stato possibile replicare la soluzione trovata.

Un esempio può essere l'andamento dell'algoritmo evoluto sulla rete 4, che trova soluzione alla generazione 99, descritto nella figura **??**. La scelta di questo esempio deriva dalla necessità di verificare quanto le stimolazioni degli individui precedenti influenzino quelle attuali. Per evitare questo genere di

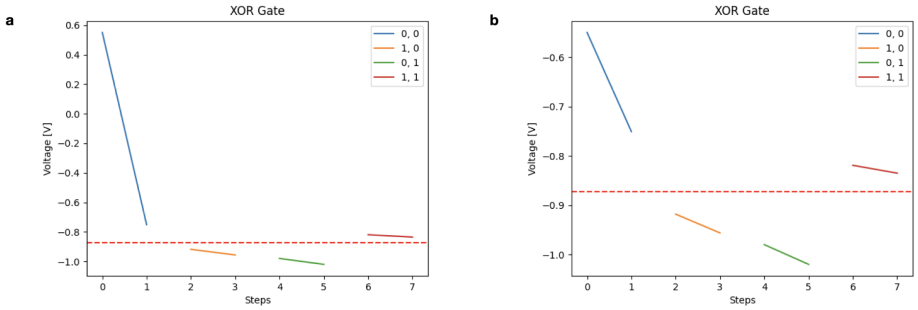

Figura 4.17: Confronto degli output ottenuti durante l'algoritmo genetico della rete 4 per le configurazioni della porta *xor* (a) durante il genetico (b) a posteriori

ripercussioni, al momento del calcolo della fitness, la rete viene, quindi, prima stimolata a *vuoto* per 200 passi, cosicché possa ritornare a uno stato *iniziale*, e solo successivamente stimolata con i valori prodotti dal genetico per altri 200 passi. Gli output prodotti dal genetico e quelli a posteriori avranno delle differenze - tolleranza fino alla quarta cifra decimale - che non influenzano la realizzazione delle porte logiche e la sua riproducibilità, come è visibile dal grafico descritto in figura 4.17.

Una volta ottenute le soluzioni ottime, si è voluto indagare l'interscambiabilità delle configurazioni trovate con altre reti. È emerso che alcune soluzioni risultano essere efficienti anche nelle reti non utilizzate durante la fase evolutiva. In particolare, nel caso della porta logica *and*, questa operazione ha reso possibile sfruttare le poche soluzioni prodotte dall'algoritmo generico su più reti e aumentare così il numero di reti configurabili. Dalla figura 4.18 alla figura 4.20 è possibile osservare quanto le configurazioni prodotte dall'evoluzione del materiale possano essere ottime soluzioni anche per le altre reti analizzate durante questi esperimenti.

L'esito di questa valutazione sembra suggerire inoltre che è possibile individuare delle reti che cambiano funzionamento a seconda della configurazione impostata, come si è visto in figura 4.16.

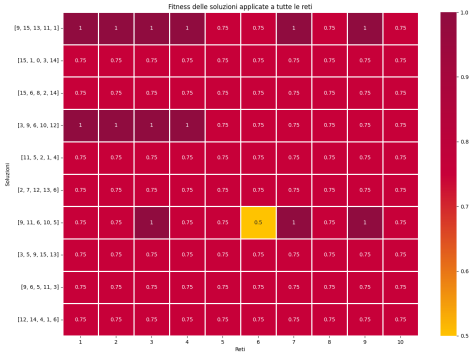

Figura 4.18: Confronto delle soluzioni ottenute per la porta *and* testate sulle altre reti

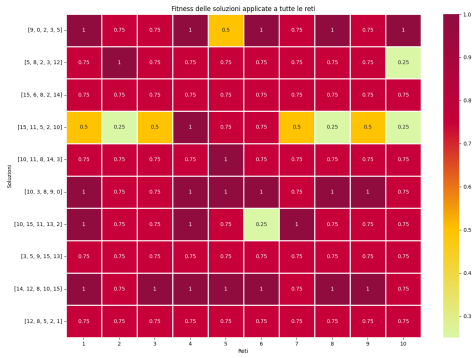

Figura 4.19: Confronto delle soluzioni ottenute per la porta *or* testate sulle altre reti

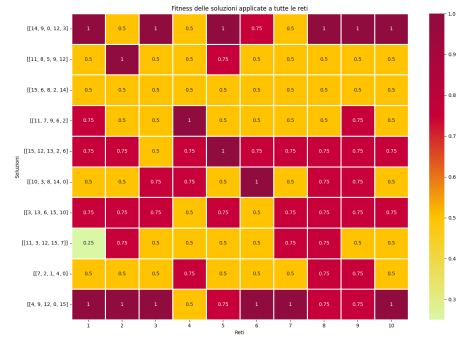

Figura 4.20: Confronto delle soluzioni ottenute per la porta *xor* testate sulle altre reti

4.1.2 Robustezza

In questa fase il lavoro si è direzionato principalmente sullo studio della porta logica *xor*. Le NN sono delle reti riconducibili a un circuito elettrico, pertanto bisogna tenere in considerazione il rumore termico prodotto dalle sue componenti. Sono state effettuate delle analisi a posteriori, per verificare la robustezza delle soluzioni trovate. La modellizzazione del rumore in elettronica avviene tramite l'utilizzo di una distribuzione gaussiana, il valore di (σ) indica quanto i valori misurati possono deviare dalla media (μ); quindi più è alta la deviazione standard maggiore sarà la variabilità o l'intensità del rumore all'interno del circuito.

Le prime analisi hanno avuto come parametri $\mu = 0$ e $\sigma = 0.01$; un rumore descritto da questi valori è la base per poter definire le soluzioni trovate robuste. Si è voluta anche esplorare una variabilità del rumore più alta, mantenendo $\mu = 0$ e aumentando σ a 0.05. Le indagini consistevano nell'applicare il rumore alle soluzioni per 30 ripetizioni così da ottenere una media dei risultati. Il grafico 4.21 mostra quanto possano essere robuste le soluzioni trovate per rete. Le soluzioni soggette a rumore con $\sigma = 0.01$ si mantengono introno al massimo e non subiscono variazioni, mentre nel caso di σ a 0.05 peggiorano di poco le loro prestazioni.

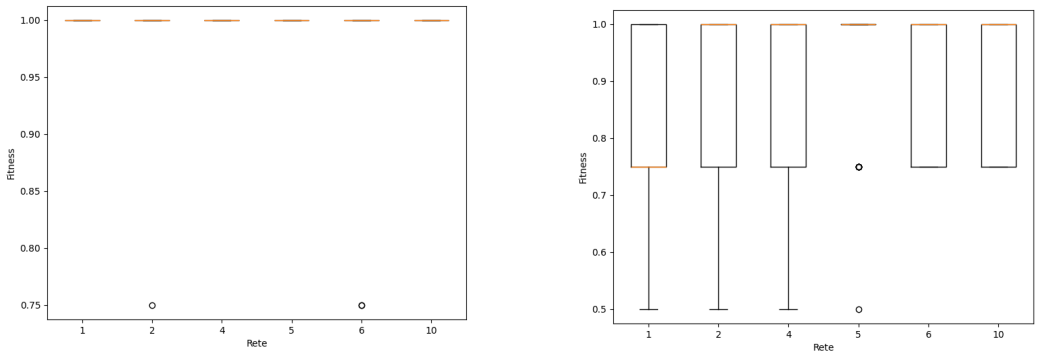

Figura 4.21: Boxplot delle 30 valutazioni con rumore avente $\sigma = 0.01$ a sinistra e $\sigma = 0.05$ a destra della porta *xor*

Per completezza sono state effettuate delle analisi di sensibilità al rumore

anche per le porte *and* e *or* riportate in figura 4.22.

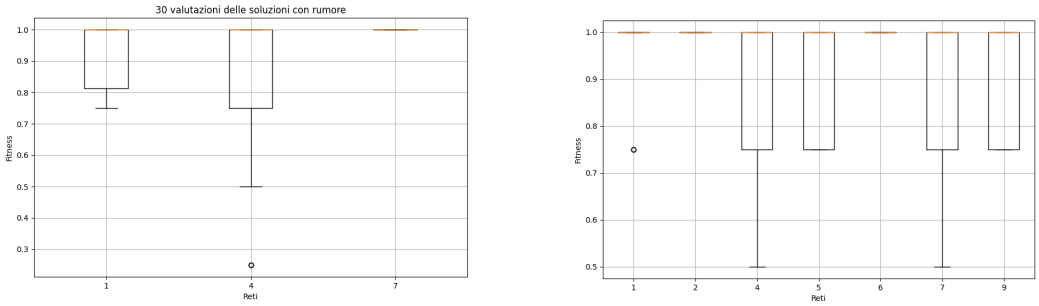

Figura 4.22: Boxplot delle 30 valutazioni con rumore avente $\sigma = 0.05$ delle porte *and* a sinistra e *xor* a destra.

Osservando questi ultimi risultati si è deciso di far evolvere il materiale tenendo in considerazione il rumore all'interno dell'algoritmo genetico, quindi inserendo la componente stocastica descritta da una distribuzione gaussiana. Il parametro di intensità di rumore scelto è pari a $\sigma = 0.05$, in quanto le soluzioni soggette a variazioni di rumore con $\sigma = 0.01$ vengono influenzate in piccolissima parte, non garantendo delle analisi rilevanti.

Nella figura 4.23 sono riportate le soluzioni valutate su 30 ripetizioni delle configurazioni ottenute mediante l'evoluzione del materiale in presenza di rumore con $\sigma = 0.05$. Per garantire la riproducibilità delle soluzioni trovate sono state effettuate 3 valutazioni durante l'algoritmo genetico per ogni individuo, ottenendo come fitness la media delle 3 singole valutazioni. Il numero di soluzioni trovate alla fine di questa evoluzione è stato lo stesso della prima evoluzione. Più o meno sono rimaste anche le stesse reti, eccetto per la rete numero 7 che ha trovato soluzione, cosa che prima faceva la rete numero 4. L'inserimento del rumore all'interno del genetico ha permesso di trovare delle soluzioni che migliorano le prestazioni in termini di sensibilità al rumore, anche se questo perfezionamento non è sempre garantito: è visibile nel caso della rete numero 5, in cui la soluzione prodotta risulta essere più sensibile al rumore.

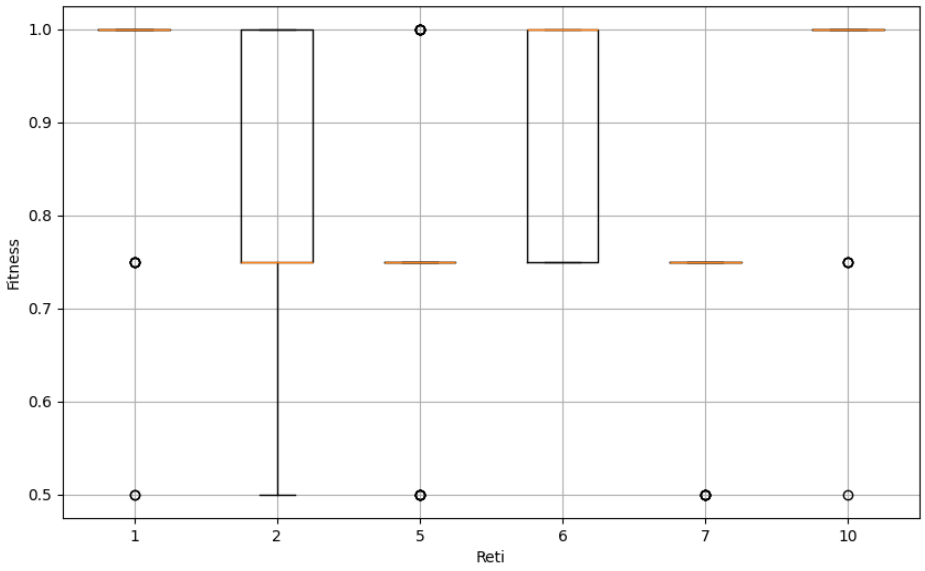

Figura 4.23: Boxplot delle 30 valutazioni dell'algoritmo genetico con rumore avente $\sigma = 0.05$ della porta *xor*.

4.1.3 Migliori soluzioni per *and*, *or* e *xor* su reti configurabili

A seguito delle considerazioni sul rumore, sono state raccolte tutte le possibili soluzioni per ogni rete e comparate in termini di robustezza. In questo modo è stato possibile configurare le reti in modo tale da eseguire tutti i compiti computazionali (*and*, *or* e *xor*) con le soluzioni migliori. Dalla figura 4.24 alla 4.28 insieme alla figura 4.16 è possibile osservare come viene stimolata la rete con le configurazioni di successo degli esperimenti condotti.

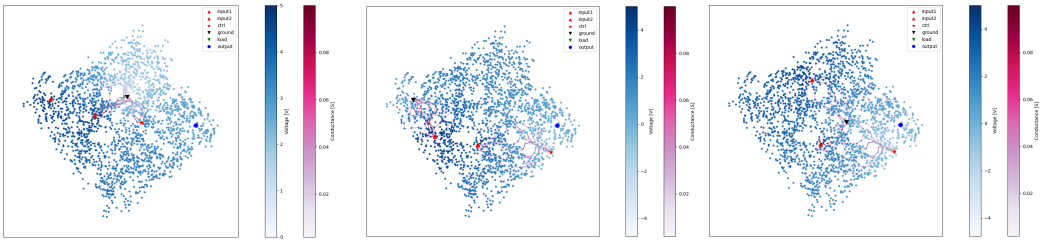

Figura 4.24: Visualizzazione grafica della stimolazione della rete 2 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 4 e 7, di ctrl 10, di ground 11 e output 13, tensione di controllo circa 3.7V, θ circa 2.17; or) pin di input 6 e 3, di ctrl 9, di ground 4 e output 13, tensione di controllo circa -4.76V, θ circa -1.74; xor) pin di input 12 e 6, di ctrl 9, di ground 10 e di output 13, tensione di controllo circa -4,76V,θ circa -1.74.

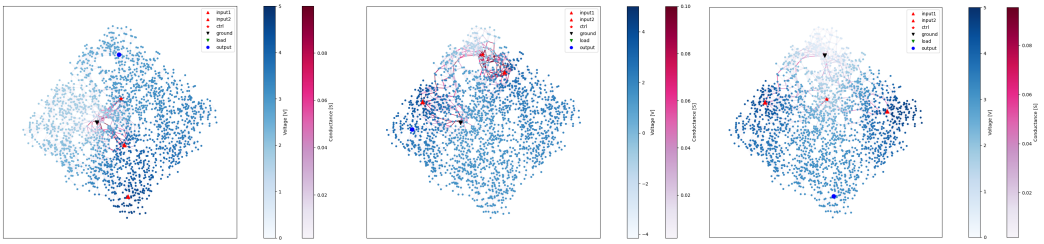

Figura 4.25: Visualizzazione grafica della stimolazione della rete 3 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 4 e 7, di ctrl 10, di ground 11 e output 13, tensione di controllo circa 3.7V, θ circa 2.17; or) pin di input 15 e 9, di ctrl 13, di ground 11 e output 16, tensione di controllo circa -4.15V, θ circa -0.03; xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5.

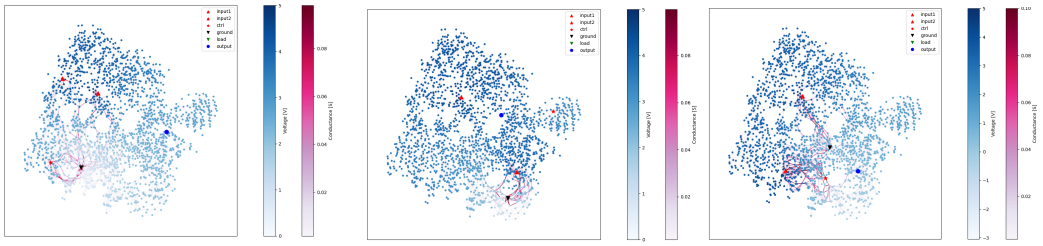

Figura 4.26: Visualizzazione grafica della stimolazione della rete 4 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 11 e 9, di ctrl 4, di ground 10 e output 1, tensione di controllo circa 2.86V, θ circa 0.7; xor) pin di input 12 e 10, di ctrl 8, di ground 7 e di output 3, tensione di controllo circa -3.0V, θ circa -0.8.

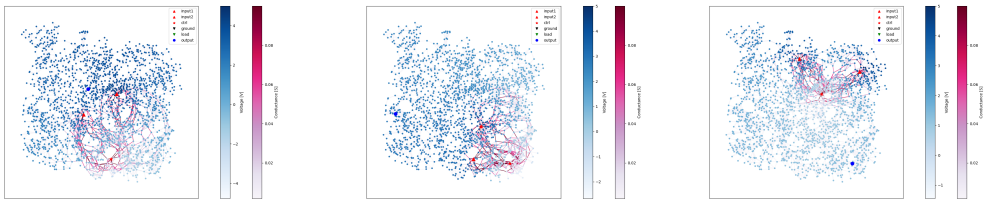

Figura 4.27: Visualizzazione grafica della stimolazione della stessa rete date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 12, di ctrl 7, di ground 11 e output 6, tensione di controllo circa 4.7V, θ circa -0.6; or) pin di input 11 e 12, di ctrl 16, di ground 14 e output 3, tensione di controllo circa 3.1V, θ circa 0.7 xor) pin di input 5 e 13, di ctrl 10, di ground 1 e di output 16, tensione di controllo circa -1.4V, θ circa -0.8.

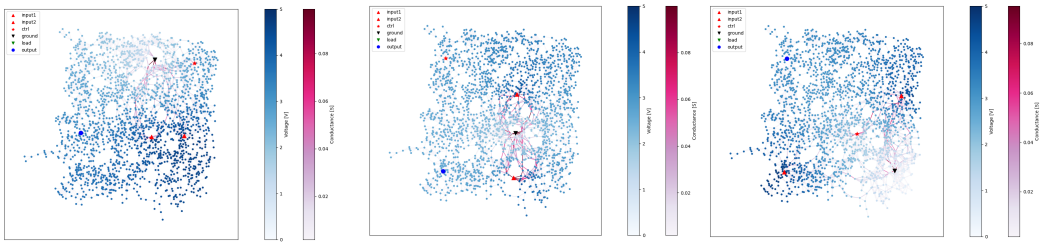

Figura 4.28: Visualizzazione grafica della stimolazione della stessa rete date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 11 e 9, di ctrl 4, di ground 10 e output 1, tensione di controllo circa 2.86V, θ circa 0.7 xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5.

4.2 Discussione

Grazie all'osservazione dei risultati ottenuti è possibile affermare che un approccio innovativo, come l'EIM, è in grado di ottenere esiti positivi nella realizzazione di porte logiche tramite l'evoluzione di NN. Più della metà delle reti prese in esame si possono configurare in modo tale che eseguano le tre porte logiche oggetto di questo studio. La ricerca di queste soluzioni è stata fatta garantendo riproducibilità - la stimolazione deve durare un numero di passi sufficiente per permettere alla rete di produrre risultati stabili - e robustezza, assicurando affidabilità in presenza di rumore con $\sigma = 0.01$. L'aggiunta di rumore durante l'evoluzione dell'algoritmo genetico ha restituito, in alcuni casi, delle soluzioni più robuste permettendo di valutare in che misure poter utilizzare questa possibile alternativa. Non è da escludere, però, la presenza di ostacoli e l'eventuale necessità di raffinare il lavoro con delle tecniche più ricercate.

4.2.1 Limiti dello studio e possibili miglioramenti

Uno degli ostacoli maggiori di questo studio è stato il tempo di esecuzione: in particolare se si considera che quello necessario per valutare un singolo individuo è pari 0.11 secondi, l'impatto maggiore sul tempo totale di esecuzione è dovuto alla simulazione della stimolazione della rete. Complessivamente, l'evoluzione di una replica nel caso peggiore, richiede un tempo pari a 15 ore. Mentre, per le porte che richiedevano una media sulle 3 valutazioni della fitness, il tempo di esecuzione si è triplicato per ogni replica (45 ore). I risultati degli esperimenti e la robustezza dei dati forniti sono stati strettamente correlati a questo problema. Questo è il motivo per cui il numero di repliche, di generazioni e di popolazione è stato giusto appena sufficiente per ottenere delle considerazioni sui risultati soddisfacenti. Per lo stesso motivo, considerazioni riguardo agli operatori dell'algoritmo genetico sono state poste in secondo piano. Ad esempio, si sarebbero potuti realizzare degli algoritmi genetici capaci di calcolare una fitness più complessa,

come il coefficiente di correlazione di Matthews (MCC) o l'F1-score, oppure migliorare le prestazioni del genetico attraverso l'aggiunta dell'operatore di crossover. In seguito si potrebbe ampliare questo studio utilizzando dei segnali in frequenza per la stimolazione della rete, che potrebbero evidenziare un'analisi più dinamica del sistema. Inoltre, si potrebbe aggiungere un'ulteriore analisi alle valutazioni con il rumore, simulando tipi di rumore diversi nel sistema tipo quelli dovuti alle conversioni analogico digitale o quelli di Flicker, dovuti a fenomeni stocastici a bassa frequenza che sono causati da imperfezioni nei materiali o nei processi di fabbricazione.

Capitolo 5

Conclusioni

In questa tesi sono state implementate le porte logiche *and*, *or* e *xor* su delle reti di nanofili neuromorfiche. I risultati ottenuti dimostrano che l'EIM è in grado di ottenere una realizzazione efficace delle porte logiche sulle NN. Sono state prodotte delle configurazioni di qualità in termini di rumore e riproducibilità, grazie all'utilizzo di un numero di passi sufficiente ad assicurare stabilità alla rete. Maggiore robustezza è stata rilevata anche con l'aggiunta di rumore durante l'evoluzione dell'algoritmo genetico. La realizzazione della porta *xor* tramite NN, essendo una funzione non separabile, ha permesso di catturare la caratteristica di non linearità di queste reti. Anche se non sono state indagate tutte le caratteristiche interessanti di questa tecnologia, gli esiti positivi in questo caso di studio hanno permesso di considerare la computazione di materia un approccio più che efficace per questi dispositivi. La progettazione delle reti non ha seguito un modello classico, bensì un modello basato sull'evoluzione diretta del materiale. La complessità strutturale dell'architettura e delle caratteristiche delle reti di nanofili sarebbero risultate estremante difficili da analizzare seguendo altri approcci: la tecnica umana di progettazione consiste nel decomporre problemi complessi in sotto problemi più comprensibili e accessibili, tecniche di questo tipo però non sempre sono applicabili e, nel caso in cui fosse possibile, non è detto che siano in grado di descrivere il materiale nella sua interezza. In conclusione, per soddisfare

la richiesta di complessità delle tecnologie emergenti nel campo del'intelligenza artificiale - dove si cerca di emulare le capacità di apprendimento e adattamento del cervello umano - può essere utile dare uno sguardo attento e farsi ispirare da ciò che è presente in natura, in quanto racchiude già in sé il percorso descritto dalle leggi evolutive in miliardi di anni.

Elenco delle Figure

1.1 Rappresentazione schematica dell'evolution in materia, presa da [13] 8

2.1 Comportamento emergente delle reti neuromorfiche auto-organizzanti a base di nanofili (NN), prese da [12]: a. Immagine al microscopio elettronico a scansione (SEM) di una rete neuromorfica auto-organizzante basata su NN altamente interconnesse (scala di 5 m) e b. dettaglio di una giunzione della NN (scala di 200 nm). c. Rappresentazione schematica del meccanismo di commutazione resistiva alle giunzioni che, sotto l'azione del campo elettrico applicato, modula la conducibilità della giunzione. . . 13

3.1 Passi di stabilizzazione sotto stimolazione: output, due ingressi e controllo. 18
3.2 Rappresentazione schematica dell'evolution in materia applicata alle NN per la realizzazione delle porte logiche 21
3.3 Descrizione dettagliata degli individui 22

4.1 Andamento della fitness della porta *xor* delle reti 1 e 2 30
4.2 Andamento della fitness della porta *xor* delle reti 3 e 4 30
4.3 Andamento della fitness della porta *xor* delle reti 5 e 6 30
4.4 Andamento della fitness della porta *xor* delle reti 7 e 8 31
4.5 Andamento della fitness della porta *xor* delle reti 9 e 10 31
4.6 Andamento della fitness della porta *and* delle reti 1 e 2 31

4.7 Andamento della fitness della porta *and* delle reti 3 e 4 32
4.8 Andamento della fitness della porta *and* delle reti 5 e 6 32
4.9 Andamento della fitness della porta *and* delle reti 7 e 8 32
4.10 Andamento della fitness della porta *and* delle reti 9 e 10 . . . 33
4.11 Andamento della fitness della porta *or* delle reti 1 e 2 33
4.12 Andamento della fitness della porta *or* delle reti 3 e 4 33
4.13 Andamento della fitness della porta *or* delle reti 5 e 6 34
4.14 Andamento della fitness della porta *or* delle reti 7 e 8 34
4.15 Andamento della fitness della porta *or* delle reti 9 e 10 34
4.16 Visualizzazione grafica della stimolazione della rete 1 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 10 e 3, di ctrl 1, di ground 4 e output 6, tensione di controllo circa 3.1V, θ circa 0.7; xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5 36
4.17 Confronto degli output ottenuti durante l'algoritmo genetico della rete 4 per le configurazioni della porta *xor* (a) durante il genetico (b) a posteriori 38
4.18 Confronto delle soluzioni ottenute per la porta *and* testate sulle altre reti 39
4.19 Confronto delle soluzioni ottenute per la porta *or* testate sulle altre reti 39
4.20 Confronto delle soluzioni ottenute per la porta *xor* testate sulle altre reti 39
4.21 Boxplot delle 30 valutazioni con rumore avente $\sigma = 0.01$ a sinistra e $\sigma = 0.05$ a destra della porta *xor* 40
4.22 Boxplot delle 30 valutazioni con rumore avente $\sigma = 0.05$ delle porte *and* a sinistra e *xor* a destra. 41

4.23 Boxplot delle 30 valutazioni dell'algoritmo genetico con rumore avente $\sigma = 0.05$ della porta *xor*. 42

4.24 Visualizzazione grafica della stimolazione della rete 2 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 4 e 7, di ctrl 10, di ground 11 e output 13, tensione di controllo circa 3.7V, θ circa 2.17; or) pin di input 6 e 3, di ctrl 9, di ground 4 e output 13, tensione di controllo circa -4.76V, θ circa -1.74; xor) pin di input 12 e 6, di ctrl 9, di ground 10 e di output 13, tensione di controllo circa -4,76V,θ circa -1.74. 43

4.25 Visualizzazione grafica della stimolazione della rete 3 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 4 e 7, di ctrl 10, di ground 11 e output 13, tensione di controllo circa 3.7V, θ circa 2.17; or) pin di input 15 e 9, di ctrl 13, di ground 11 e output 16, tensione di controllo circa -4.15V, θ circa -0.03; xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5. 43

4.26 Visualizzazione grafica della stimolazione della rete 4 date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 11 e 9, di ctrl 4, di ground 10 e output 1, tensione di controllo circa 2.86V, θ circa 0.7; xor) pin di input 12 e 10, di ctrl 8, di ground 7 e di output 3, tensione di controllo circa -3.0V, θ circa -0.8. 44

4.27 Visualizzazione grafica della stimolazione della stessa rete date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 12, di ctrl 7, di ground 11 e output 6, tensione di controllo circa 4.7V, θ circa -0.6; or) pin di input 11 e 12, di ctrl 16, di ground 14 e output 3, tensione di controllo circa 3.1V, θ circa 0.7 xor) pin di input 5 e 13, di ctrl 10, di ground 1 e di output 16, tensione di controllo circa -1.4V, θ circa -0.8. 44

4.28 Visualizzazione grafica della stimolazione della stessa rete date configurazioni diverse con tensioni di ingresso entrambe a 5V, in ordine: and) pin di input 10 e 14, di ctrl 16, di ground 12 e output 2, tensione di controllo circa 3.1V, θ circa 0.2; or) pin di input 11 e 9, di ctrl 4, di ground 10 e output 1, tensione di controllo circa 2.86V, θ circa 0.7 xor) pin di input 15 e 1, di ctrl 10, di ground 13 e di output 4, tensione di controllo circa 1.3V, θ circa 0.5. 45

Elenco delle Tabelle

3.1 Iperparametri del genetico 24
3.2 Scheda tecnica delle NN 25

4.1 Risultati dell'algoritmo genetico 35

Bibliografia

[1] Andrew Adamatzky, Ben De Lacy Costello, and Takashi Asai. *Reaction-Diffusion Computers*. Elsevier, Amsterdam, 2005.

[2] Paolo Baldini. *Online Adaptation of Robots Controlled by Nanowire Networks*. Master's thesis, Alma Mater Studiorum - Università Di Bologna, Cesena, Italia, 2020-2021.

[3] Paolo Baldini, Andrea Roli, and Michele Braccini. On the performance of online adaptation of robots controlled by nanowire networks. *IEEE Access*, (99):1–1. doi: 10.1109/ACCESS.2023.3345224. URL `https://doi.org/10.1109/ACCESS.2023.3345224`.

[4] Hajo Broersma. *Evolution in Nanomaterio: The NASCENCE Project*, pages 67–82. Springer, 2018. doi: 10.1007/978-3-319-67997-6_4.

[5] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, July 1985. doi: 10.1098/rspa.1985.0070. URL `https://doi.org/10.1098/rspa.1985.0070`.

[6] A. E. Eiben, S. Kernbach, and E. Haasdijk. Embodied artificial evolution. *Evolutionary Intelligence*, 5(4):261–272, December 2012. doi: 10.1007/s12065-012-0071-x. URL `https://doi.org/10.1007/s12065-012-0071-x`.

[7] Pask Gordon. Physical analogues to the growth of a concept. in: Mechanisation of thought processes. *National Physical Laboratory Symposium*, pages 877–922, 1958.

[8] John Henry Holland. *Adaptation in Natural and Artificial Sy- stems, An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Cambridge, MA, 1992.

[9] Gianluca Milano and altri. In materia implementation strategies of physical reservoir computing with memristive nanonetworks. *Journal of Physics D: Applied Physics*, 56(8), 2023. doi: 10.1088/1361-6463/acb7ff. URL `https://doi.org/10.1088/1361-6463/acb7ff`.

[10] Gianluca Milano, Giacomo Pedretti, Matteo Fretto, Luca Boarino, Fabio Benfenati, Daniele Ielmini, Ilia Valov, and Carlo Ricciardi. Brain-inspired structural plasticity through reweighting and rewiring in multi-terminal self-organizing memristive nanowire networks. *Advanced Intelligent Systems*, 2020. doi: 10.1002/aisy.202000096. URL `https://doi.org/10.1016/j.neunet.2022.02.022`.

[11] Gianluca Milano, Enrique Miranda, and Carlo Ricciardi. Connectome of memristive nanowire networks through graph theory. *Neural Networks*, 2022. doi: 10.1016/j.neunet.2022.02.022. URL `https://doi.org/10.1016/j.neunet.2022.02.022`.

[12] Gianluca Milano, Fabio Michieletti, Carlo Ricciardi, and Enrique Miranda. Self-organizing neuromorphic nanowire networks are stochastic dynamical systems. *Research Square*, April 2024. doi: 10.21203/rs.3.rs-4102090/v1. URL `https://doi.org/10.21203/rs.3.rs-4102090/v1`. This work is licensed under a Creative Commons Attribution 4.0 International License.

[13] Julian F. Miller, Simon L. Harding, and Gunnar Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*,

7(2):49–67, April 2014. doi: 10.1007/s12065-014-0106-6. URL https://doi.org/10.1007/s12065-014-0106-6.

[14] DEAP Development Team. Deap documentation, 2024. URL https://deap.readthedocs.io/en/master/. Ultimo accesso: 27 giugno 2024.

[15] C. A. Jr. Thomas, P. A. Springer, G. E. Loeb, Y. Berwald-Netter, and L. M. Okun. A miniature microelectrode array to monitor the bioelectric activity of cultured cells. *Experimental Cell Research*, 74(1):61–66, 1972. doi: 10.1016/0014-4827(72)90481-8. URL https://doi.org/10.1016/0014-4827(72)90481-8.

[16] Adrian Thompson, Inman Harvey, and Philip Husbands. Unconstrained evolution and hard consequences. In Eduardo Sanchez and Marco Tomassini, editors, *Towards Evolvable Hardware*, pages 136–165, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-49947-3.

[17] Ruomin Zhu, Sam Lilak, Alon Loeffler, Joseph Lizier, Adam Stieg, James Gimzewski, and Zdenka Kuncic. Online dynamical learning and sequence memory with neuromorphic nanowire networks. *Nature Communications*, 14(1):6697. doi: 10.1038/s41467-023-42470-5. URL https://doi.org/10.1038/s41467-023-42470-5.

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a tutte le persone che mi hanno sostenuto e guidato lungo il percorso della realizzazione di questa tesi. Innanzitutto, vorrei ringraziare il mio supervisore, Andrea Roli, e i miei correlatori, Paolo Baldini e Michele Braccini, per la guida preziosa, il sostegno e i consigli durante tutto il periodo di ricerca. Senza il loro impegno e la loro competenza, questo lavoro non sarebbe stato possibile. Vorrei inoltre ringraziare tutti i miei colleghi e amici che hanno condiviso con me idee e discussioni stimolanti, contribuendo così al miglioramento di questo lavoro. Un grazie di cuore va anche alla mia famiglia per il loro amore, il loro sostegno incondizionato e per aver creduto in me in ogni fase del mio percorso accademico.

Grazie a tutti coloro che hanno reso possibile questo traguardo con il loro contributo diretto o indiretto.