ALMA MATER STUDIORUM – UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA INFORMATICA

TITOLO DELLA TESI

Modularità: dai Sistemi Biologici ai Sistemi Artificiali

Tesi in

Intelligenza Artificiale Ls

Relatore

Andrea Roli

Presentata da

Paola Castrignano

Sessione I

Anno accademico 2013/2014

Indice

Introduzione

I sistemi di intelligenza artificiale vengono spesso messi a confronto con gli aspetti biologici riguardanti il cervello umano.
L'interesse per la modularità è in continua crescita, che sta portando a risultati davvero interessanti attraverso l'utilizzo di sistemi artificiali intelligenti, come le reti neuronali. Molte reti, sia biologiche sia artificiali sono organizzate in moduli, i quali rappresentano *cluster* densi di parti interconnesse fra loro all'interno di una rete complessa.
Nel campo dell'ingegneria, si usano *design* modulari per spiegare come una macchina è costituita da parti separate.
Lo studio della struttura e delle funzioni di organismi/processi complessi si basa implicitamente su un principio di organizzazione modulare, vale a dire si dà per acquisito il fatto che siano modulari, cioè composti da parti con forma e/o funzioni diverse.
Questo elaborato si propone di esporre gli aspetti fondamentali riguardanti la modularità di reti neuronali, le sue origini evolutive, le condizioni necessarie o sufficienti che favoriscono l'emergere dei moduli e relativi vantaggi.
Il primo capitolo fornisce alcune conoscenze di base che permettono di leggere gli esperimenti delle parti successive con consapevolezza teorica più profonda. Si descrivono reti neuronali artificiali come sistemi intelligenti ispirati alla struttura di reti neurali biologiche, soffermandosi in particolare sulla rete *feed-forward*, sull'algoritmo di *backpropagation* e su modelli di reti neurali modulari.
Il secondo capitolo offre una visione delle origini evolutive della modularità e dei meccanismi evolutivi riguardanti sistemi biologici, una classificazione dei vati tipi di modularità, esplorando il concetto di modularità nell'ambito della psicologia cognitiva. Si analizzano i campi disciplinari a cui questa ricerca di modularità può portare vantaggi.
Dal terzo capitolo che inizia a costituire il corpo centrale dell'elaborato, si dà importanza alla modularità nei sistemi computazionali, illustrando alcuni casi di studio e relativi metodi presenti in letteratura e fornendo anche una misura quantitativa della modularità.
Si esaminano le varie possibilità di evoluzione della modularità: spontanea, da specializzazione, duplicazione, *task-dependent,* ecc. passando a emulare l'evoluzione di sistemi neurali modulari con applicazione al noto modello "What-Where" e a vari modelli con caratteristiche diverse.
Si elencano i vantaggi che la modularità produce, soffermandosi sull'algoritmo di apprendimento, sugli ambienti che favoriscono l'evoluzione della modularità con una serie di confronti fra i vari tipi, statici e dinamici. In ultimo, come il vantaggio di avere connessioni corte possa portare a sviluppare modularità.
L'obiettivo comune è l'emergere della modularità in sistemi neuronali artificiali, che sono usati per applicazioni in numerosi ambiti tecnologici.

CAPITOLO 1

Reti Neurali Artificiali [*]

Le Reti Neurali Artificiali (ANN) è quello che più di ogni altro ha abbracciato questa filosofia.
In Intelligenza Artificiale si fa riferimento a un'ispirazione biologica ritenendo che l'intelligenza "naturale" del cervello possa esser riprodotta anche solo in parte e in un contesto simulativo tramite calcolatore.
La distanza fra modello artificiale e biologico è evidente soprattutto nel momento in cui si tenta di riprodurre su calcolatore capacità plastiche del sistema nervoso umano.
La semplicità dell'architettura neuronale artificiale ha ridotto la variabilità del comportamento cellulare: tutti i fenomeni di natura biochimica e bioelettrica che avvengono nella comunicazione sinaptica sono stati riassunti in un numero chiamato "peso sinaptico" il quale modula il segnale proveniente dall'assone del neurone eccitato. L'attività di neurotrasmettitori e ricettori nella comunicazione sinaptica è variabile e questa variabilità gioca un ruolo importante nei fenomeni di apprendimento. Per tale motivo è emersa la necessità di algoritmi efficienti aventi capacità di adattamento. La soluzione più brillante adottata è stata l'idea della retropropagazione dell'errore (*backpropagation*) che ha permesso di regolare i pesi, prendendo spunto dall'architettura di rete neuronale. L'algoritmo di backpropagation, oltre a fornire un incremento delle capacità di apprendimento delle ANN, ha accentuato anche le distanze fra modelli neurali biologici e artificiali: su reti *multilayer* rappresenta uno strumento matematico-statistico di approssimazione e ottimizzazione in sistemi non lineari, ma uno svantaggio se si vuole riprodurre l'attività intelligente del cervello.
Vi sono stati vari tentativi di produrre algoritmi di apprendimento biologicamente plausibili. Sono emersi alcuni esempi in cui questo problema sembrerebbe superato e quasi tutte le ipotesi fanno capo a un unica regola di modulazione dell'adattamento della risposta sinaptica: un esempio importante è la "regola di Hebb". Questa regola rappresenta un modello "ponte" fra apprendimento biologico e artificiale e descrive la capacità di adattamento delle unità neuronali in contesti di apprendimento associativo. La sua formulazione originale risale al 1949 ed è fornita dalla seguente assunzione (teorica) da Donald Hebb:

quando un assone della cellula A (la cellula presinaptica) prende parte ripetutamente al processo di eccitamento della cellula B (la cellula postsinaptica), qualche cambiamento strutturale o metabolico subentra in una o entrambe le cellule in modo che l'efficienza di A come cellula eccitatrice di B aumenti.

[*] Gli argomenti illustrati in questo capitolo sono prevalentemente tratti dai seguenti riferimenti:
Dynamics of complex systems, Neural Networks I: Subdivision and Hierarchy, Bar-Yam, Addison-Wesley, Cap. 2, 1997.
Modularity: Understanding the development and evolution of complex natural systems a cura di W. Callebaut, D. Rasskin-Gutman, Cambridge, The MIT Press, Cap. 3, 2005.
Landassuri-Moreno, J.A. Bullinaria "A New Approach for Incremental Modular Neural Networks in Time Series Forecasting" in Proceedings of the Workshop on Computational Intelligence, UK, 2008.
www.semeion.it

Ulteriore contributo di Hebb fu quello di espandere tale regola a gruppi di neuroni:

eccitandosi insieme formano raggruppamenti cellulari associativi.

Secondo Hebb, l'attività cognitiva è determinata da un attivazione sequenziale di questi insiemi. La più semplice delle formulazioni matematiche proposte è:

$$\Delta w_{ij} = \eta x_i x_j$$

Δw_{ij} rappresenta la variazione dell'efficienza sinaptica fra neurone i e neurone j; η la velocità di adattamento.
Se le due cellule si attivano insieme, l'efficienza della sinapsi che le collega viene rafforzata dando luogo ad un fenomeno di adattamento associativo, il quale prescinde dalla presenza di una supervisione esterna all'attività neuronale.

1.1

Analogie biologiche e storia delle ANN

Il modello costitutivo di un ANN è ispirato ai fondamenti biologici riguardanti tessuti nervosi del cervello umano, il quale contiene neuroni (detti anche cellule nervose). Ogni neurone viene distinto in tre parti: un insieme di fibre di ingresso (dendriti), un corpo cellulare (soma) e una diramazione principale (assone). Gli assoni, a loro volta, si dividono in diverse terminazioni, ciascuna delle quali a contatto con altri neuroni. Ogni cellula nervosa può ricevere sino a 10.000 impulsi in ingresso. Le zone di contatto e di trasmissione degli impulsi sono chiamate sinapsi. Le controparti artificiali di neuroni e sinapsi sono le unità computazionali e le interconnessioni. I neuroni possono generare impulsi di tipo elettrico i quali vengono trasmessi attraverso l'assone e verso le sinapsi. Ogni impulso può contribuire all'eccitazione o all'inibizione del neurone ricevente. Inoltre l'efficienza con cui le sinapsi trasmettono i segnali può variare nel tempo.
Nel cervello i neuroni sono organizzati in vere e proprie reti, a loro volta incluse in moduli indipendenti gli uni dagli altri. Tale fattore assicura la possibilità di elaborare dati in parallelo e operare più processi simultaneamente.
Il modello artificiale di trasmissione delle informazioni fra i vari nodi di un ANN risulta una semplificazione di tali impulsi di sistemi biologici.

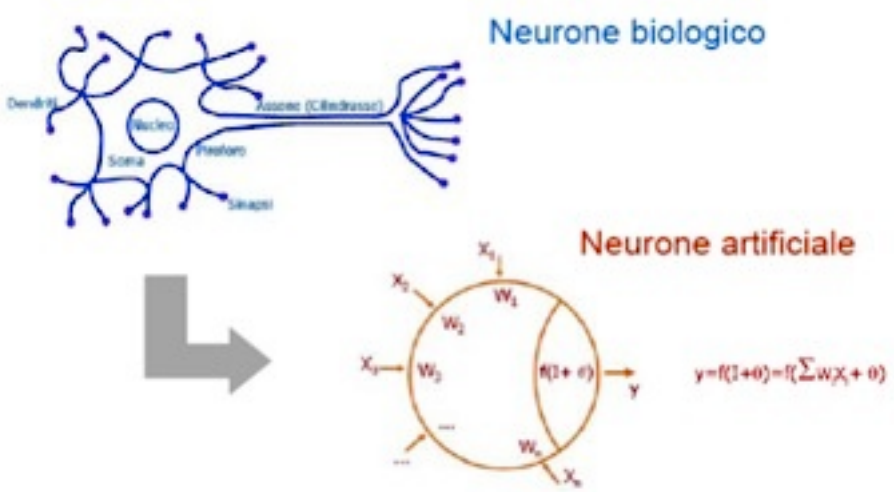


Fig. 1. Modello schematico di neurone biologico e neurone artificiale

Lo studio delle ANN risale al 1927, ma più tardi nel 1943 il neuroscienziato McCulloch con la collaborazione del logico Pitts propose il primo modello di neurone artificiale nell'articolo *"A logical calculus of the ideas immanent in nervous activity*". In tale modello i nodi producono risultati binari e ci può essere un numero illimitato di input, distinti in inibitori o eccitatori, a seconda delle connessioni di trasmissione. Se anche uno solo degli input inibitori è 1, l'unità neurale viene inibita e il risultato della computazione è 0. Se ciò non avviene si computa l'eccitazione totale x, cioè la somma degli input eccitatori e, si confronta x con un valore di soglia θ proprio del neurone artificiale. Se $x >= \theta$ l'unità riporta in uscita 1, altrimenti il risultato è ancora 0.
La prima rete neurale che conobbe notevole successo commerciale fu il percettrone, proposto da Rosenblatt nel 1958. L'innovazione di tale modello fu l'introduzione di pesi diversi per diverse connessioni e una speciale struttura di interconnessione. Oggi viene impiegato nel riconoscimento di forme in problemi di classificazione. Si tratta di una rete formata da un sottoinsieme, detto strato di unità di input e da uno strato di unità di output. Non ci sono connessioni fra unità di uno stesso strato e le connessioni fra unità di input e output sono unidirezionali.
Le reti che non contengono connessioni fra unità di uno stesso strato e in cui la propagazione dei dati è diretta esclusivamente dalle unità di ingresso a quelle di uscita sono dette reti *feedforward*. L'informazione è contenuta in ciascun nodo e computata tramite n linee di input attraverso una funzione di integrazione:

$$\Psi : R^n \rightarrow R$$

Nel percettrone è una somma di input.
L'eccitazione totale delle unità computazionali è valutata con una funzione di attivazione:

$$\Phi : R \rightarrow R$$

Nel percettrone compare tale somma con un valore di soglia. L'apprendimento del percettrone avviene in maniera iterativa, stabilendo dall'esterno se la risposta fornita dalla rete a fronte di un certo input, sia corretta o errata. In questo secondo caso, si modificano i pesi delle interconnessioni in modo che la rete si avvicini alla risposta corretta.
Nel 1969, i matematici Minsky e Papert analizzarono limiti e possibilità del percettrone semplice (con un unico strato di pesi da adattare) mostrando che la classe di funzioni in grado di discriminare è limitata a quelle delle funzioni linearmente separabili.
Quindi è possibile distinguere due tipi di reti neurali artificiali in grado di modellare reti biologiche o applicazioni di *pattern recognition* [BAY97]: il primo tipo è chiamata rete attrattore (Fig. 2 sinistra) e consiste di neuroni matematici identificati come variabili che rappresentano l'attività. I neuroni sono collegati da sinapsi costituite da variabili rappresentanti la forza della sinapsi fra i due neuroni e devono essere simmetriche; il secondo tipo (Fig. 2 destra) è chiamata rete feed-forward. Una seconda distinzione fra i due tipi di rete riguarda la scelta di rappresentazione dell'attività neurale. La rete attrattore usa variabili binarie, mentre la rete feed-forward un numero reale in un intervallo limitato. Tali scelte dipendono dalla risposta non lineare dei neuroni. La sua attività in un dato tempo è pensata per essere una funzione sigmoidale dell'influenza di altri neuroni. Quindi vuol dire che a livelli moderati di eccitazione, l'attività è proporzionale all'eccitazione. Mentre per livelli alti di eccitazione l'attività satura. La rete feed-forward usa un modello di riposta non lineare che include entrambi, regime di saturazione e lineare. Mentre la rete attrattore rappresenta solo il regime di saturazione. Generalizzando quest'ultima a includere il regime lineare, non porta a cambiamenti importanti dei risultati. Viceversa, entrambi i regimi sono necessari per la rete feed-forward per essere un modello significativo.

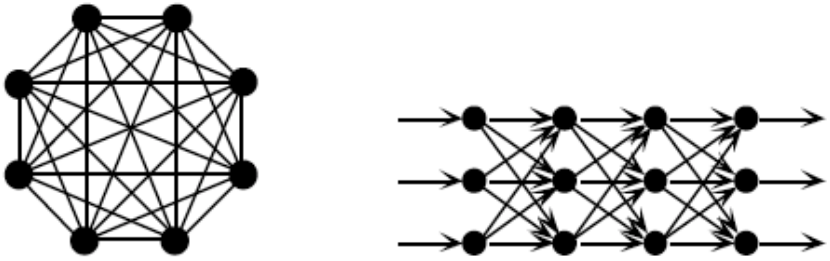


Fig. 2. Rete attrattore, a sinistra. Rete feed-forward, a destra.

1.2

Connessioni, complessità e adattamento

Le reti neurali artificiali costituiscono un approccio computazionale alternativo agli algoritmi.
Una rete neurale artificiale è pensata come un grafo orientato pesato dove i nodi elaborano l'informazione e prendono il nome di unità computazionali, agenti o neuroni artificiali, gli archi vengono chiamati sinapsi, interconnessioni o connessioni, le quali trasmettono dati numerici da nodo a nodo e memorizzano i risultati di una specifica computazione. La definizione di grafo caratterizza in parte l'architettura della rete, poiché ne determina la coppia (V, E) di nodi e connessioni. L'unità fondamentale è il neurone artificiale (Fig. 3). Può ammettere in ingresso n canali di input, ciascuno in grado di trasmettere un valore reale generico x_i. La funzione primitiva o funzione di attivazione f computata nel corpo del neurone artificiale può essere selezionata arbitrariamente. I canali di input hanno un peso associato e l'informazione in ingresso x_i è moltiplicata per il corrispondente peso w_i. L'informazione trasmessa è integrata nel neurone, il quale esegue una somma dei prodotti $x_i\, w_i$ e viene valutata la funzione di attivazione.

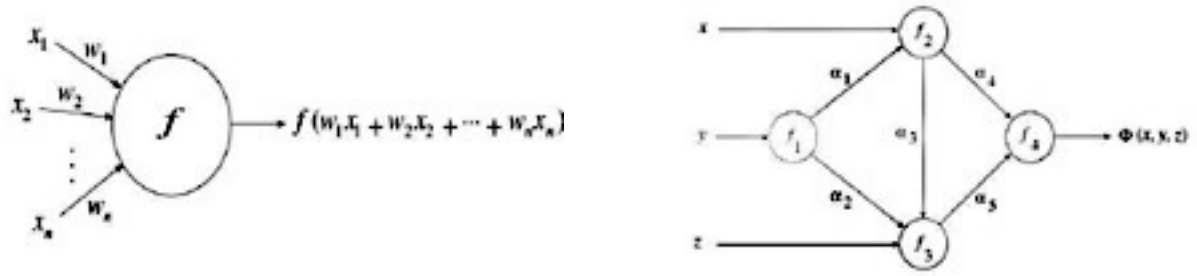


Fig. 3. Schematizzazione di un neurone artificiale, a sinistra. Schematizzazione di una generica rete neurale artificiale, a destra.

Una rete generica, come quella rappresentata può essere pensata come una funzione Φ che viene valutata nel punto (x, y, z). I nodi implementano funzioni primitive f_1, f_2, f_3, f_4 che vengono combinate per produrre Φ funzione di rete o *network function*. Pesi diversi α_1, α_2, α_3, α_4, α_5 producono funzioni di rete diverse.
Gli elementi importanti in una qualsiasi architettura ANN sono: la struttura dei nodi, le funzioni primitive arbitrarie, la topologia e l'algoritmo di apprendimento per trovare i pesi.
Alla base di ogni rete neurale artificiale vi è un' ipotesi: l'intelligenza del sistema globale non

implica necessariamente l'intelligenza dei suoi singoli componenti. L'attenzione è spostata dall'aspetto microscopico ai "fenomeni emergenti", come si presenta il sistema nella sua interezza, ciò che non può essere previsto o dedotto dal funzionamento di singole unità (elemento caratterizzante dei sistemi complessi). Tale emergenza è importante e anche la capacità di adattamento di una ANN all'ambiente circostante.
La capacità di adattamento è indice di una relazione specifica fra il sistema stesso e il suo ambiente, dove il primo è in grado di apprendere dal secondo senza una programmazione specifica.

1.3

Apprendimento

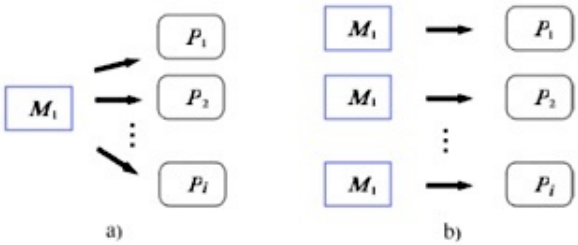
L'algoritmo di apprendimento si occupa di minimizzare l'errore in uscita e consiste nel trovare elementi non noti di una data architettura, come la funzione di attivazione delle unità computazionali o i pesi delle interconnessioni partendo da un insieme di coppie di vettori di input e di output desiderati. La dimensione del problema dipende dal numero di variabili sconosciute da determinare e dal tipo di algoritmo scelto per l'individuazione.
E' stato dimostrato che l'apprendimento di una rete con solo tre unità può essere NP-completo [NAT91].
Esistono tipologie diverse di apprendimento: apprendimento con rinforzo dove la rete si comporta come un agente in grado di imparare dall'ambiente circostante per mezzo di un processo di prova ed errore; apprendimento non supervisionato, in cui la rete non riceve alcuna informazione dall'ambiente sui valori desiderati in risposta, ma estrapola autonomamente i dati di input e acquisisce la possibilità di individuare correlazioni in nuovi insiemi di dati; apprendimento supervisionato mira a costruire un sistema in modo da consentirgli di risolvere compiti in maniera autonoma sulla base di una serie di esempi ideali, costituiti da coppie di input e di output desiderati forniti in partenza.

1.4

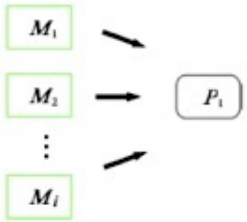
ANN e moduli

Le reti neurali artificiali risolvono un insieme di problemi, ciascuna ha la caratteristica di risolverne uno [MBU08]. L'approccio classico è quello di un modulo o ANN M_1 che risolve un problema P_1, quindi una rete o modulo specializzati per risolvere un solo problema.

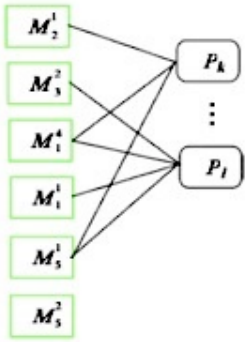
Un altro approccio è quello di un modulo o ANN M_1 che risolve diversi problemi P_i : a) La rete M_1 ha n output, ognuno risolve un problema P_i ; b) la rete ha solo un output ma risolve P_i task, per es. è addestrata con tutti i pattern di apprendimento di ogni problema.



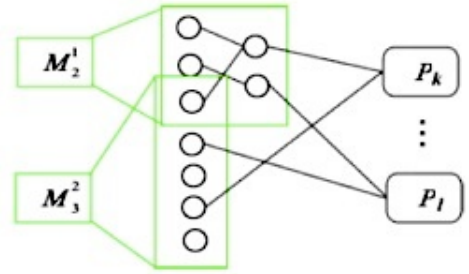
Se i task sono simili o strettamente correlati nel loro comportamento, l'interferenza *cross-task* è minima ed è possibile applicare gli approcci a) e b). Per problemi non correlati l'interferenza *cross-task* potrebbe essere alta, meglio avere moduli indipendenti per ciascun problema. Anche in un singolo task potrebbero esserci diversi comportamenti corrispondenti a input diversi o intervalli di tempo, per cui è meglio avere moduli indipendenti.
Tuttavia, ci sono procedure diverse per risolvere un problema con *Modular Neural Networks* (MNN), ad es. il metodo *Ensemble* o l'approccio *divide et impera*.
Una rete neurale modulare (MNN) con M_i moduli diversi per risolvere un problema P_1.



L'obiettivo *Incremental Modular Neural Networks* è quello di studiare MNN con rappresentazioni più grandi che permettono ai moduli esistenti di essere riusati per risolvere nuovi problemi. Si possono avere: moduli in grado di risolvere nuovi problemi della stessa classe ma come problemi esistenti; moduli in grado di collaborare fra loro per risolvere nuovi problemi in modo da scoprire una procedura evolutiva; moduli creati per risolvere un nuovo problema unico, casi in cui moduli esistenti non possono collaborare fra loro per risolvere il problema in questione. Il processo evolutivo mira a garantire che il sistema risultante abbia un architettura compatta e incrementale.
Il primo approccio MNN risulta il più semplice per sviluppare una MNN con queste caratteristiche: il super-indice j nei moduli $M_i^{\ j}$ rappresenta il task per cui è stato creato e il sotto-indice i rappresenta il numero del modulo per il suo task originario.



Si può andare più nello specifico usando insieme moduli esistenti per risolvere ogni nuovo task ma riusare sottoinsiemi di neuroni all'interno dei moduli come nell'approccio seguente: inizialmente i moduli esistenti risolvono un solo task (approccio classico) ma come l'architettura inizia a crescere, più moduli o neuroni possono essere condivisi e contribuire a risolvere nuovi task. Si scoprono nuovi moduli per i nuovi task al loro arrivo.



1.5

Modelli di ANN: *Modular Neural Network*

Modelli di rete neurali artificiali modulari consentono di affrontare problemi complessi attraverso la scomposizione in problemi semplici di soluzione più immediata.
La sfida riguarda non il miglioramento delle tecniche di apprendimento ma la creazione di moduli di reti che armonizzino la competizione e cooperazione dei singoli elementi.
L'apprendimento di *Modular Neural Networks (MNN)* può essere più veloce di reti singole. Le prime non risentono di problemi di interferenza "*temporal* e *spatial crosstalk* " che tendono a ritardare l'apprendimento delle seconde. L'interferenza s*patial crosstalk* avviene quando ad una rete si fanno apprendere contemporaneamente due task diversi. Le difficoltà di apprendimento possono dipendere dal fatto che le unità di uscita forniscono informazioni contradditorie sull'errore alle unità nascoste. Nel caso di interferenza *temporal crosstalk*, le informazioni contrastanti non sono fornite nello stesso istante di tempo ma in momenti successivi, ad esempio nel caso in cui si voglia addestrare una rete su due problemi in due momenti diversi. L'apprendimento del secondo problema comprometterà l'apprendimento del primo. Quindi si può stabilire che reti neurali modulari apprendono più velocemente perché funzioni simili possono essere apprese dallo stesso modulo sfruttando gli effetti di un trasferimento positivo di apprendimento. Funzioni diverse invece possono essere apprese da reti diverse, evitando gli effetti negativi delle interferenze.
La modularità consente di inserire in una rete, conoscenze a priori sul problema in esame, le quali risultano importanti per favorire l'apprendimento.

1.5.1

Mixture of Experts

La *Mixture of Experts* (ME) è un'architettura di reti neurali modulari proposta da Jacobs e al., nel 1991, costituita da N moduli, generalmente reti *feed forward multilayer* dette *Expert Networks* (EN) e una rete di controllo *Gating Network* (GN) (Fig. 4).

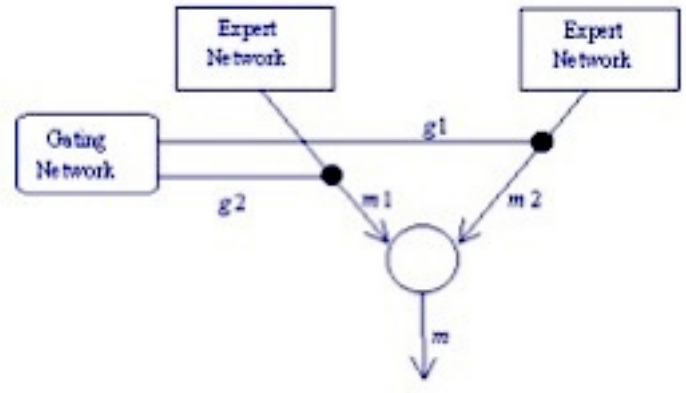


Fig. 4. Mixture of Experts (ME).

Queste reti sono addestrate simultaneamente in modo che le EN competano per apprendere pattern d'ingresso. Tale competizione è mediata da GN, la quale produce i coefficienti *g* che servono per pesare i contributi delle EN. Il modo più semplice per affrontare la ME è quello probabilistico. Le EN sono viste come processi statistici che generano un output *y* con probabilità condizionata all'ingresso *x* e al valore dei pesi dell'EN considerata. La GN invece, è vista come un classificatore che mappa l'ingresso *x* sulla probabilità che le varie EN siano in grado di generare l'output desiderato. La GN suddivide lo spazio di ingresso in regioni che corrispondono alle varie EN assegnando un vettore di probabilità [*g1*.... *gN*] ad ogni punto dello spazio di ingresso, cioè genera una suddivisione dello spazio di ingresso consentendo ai dati di appartenere simultaneamente a più regioni e ai parametri di una regione di essere influenzati dai dati di regioni attigue.
Nel 1994, Jordan e Jacobs proposero un'evoluzione dell' ME: l'architettura *Hierarchical Mixture of Experts* (HME) (Fig. 5). L'idea di base è quella di trattare ogni EN come una nuova ME ricorsivamente. Da un approccio di questo tipo, risulta una struttura ad albero in cui le GN sono ai nodi non terminali e generano una suddivisione ricorsiva dello spazio di ingresso e le EN sono le foglie che producono un uscita che viene poi "pesata" dalle GN per ricostruire l'uscita.

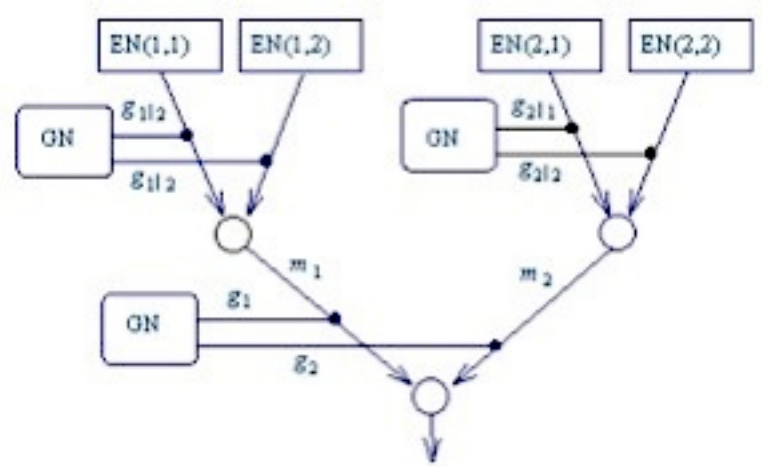


Fig. 5. Hierarchical Mixture of Experts (HME).

CAPITOLO 2

Origini Evolutive della Modularità [*]

Nella biologia dello sviluppo evolutivo è importante la relazione che intercorre fra sviluppo ed evoluzione. La sfida proposta è quella di integrare lo sviluppo di genomi e organismi e come la popolazione si approccia all'evoluzione.
La connessione non risulta semplice: si tenta di mettere in risalto la dinamica delle popolazioni, la fitness, l'ottimizzazione funzionale e la selezione naturale come connessione cardine fra le due.
Wagner, Mezey e Calabretta [WMC01] hanno fornito una panoramica di modelli di origine evolutiva dei moduli, identificando fino a otto diversi possibili meccanismi e discutendo ulteriori problemi coinvolti con una spiegazione causale dell'origine della modularità. La loro visione sulla modularità resta sulla struttura della mappa genotipo–fenotipo implicando che specifici insiemi di geni sono integrati in maniera forte. Suggeriscono che l'origine dei moduli è un caso speciale di evoluzione dell'architettura genetica e come tale, qualsiasi modello evolutivo deve spiegare come la selezione naturale possa produrre questa distribuzione di effetti genetici. In più che sarebbe illusorio ritenere che un meccanismo spieghi l'origine della modularità in tutte le circostanze.
Sull'origine di moduli evolutivi o "moduli di selezione", Brandon (1999) rivede prima i precedenti lavori di Lewontin (1974) sulla "quasi indipendenza" dei caratteri che consente alla selezione di agire su una caratteristica senza alterare le altre. Bonner (1988) in alternativa crede che i moduli possano esser dedotti "indirettamente" da dati o basati su osservazione "diretta". Brandon sostiene che un modulo evolutivo debba essere funzionale e di sviluppo modulare, contrappone la propria "analisi concettuale" con le "ipotesi empiriche" del modello di Wagner–Altenberg (1996).
I moduli interagiscono in maniera meccanica o competono in maniera selettiva? Oppure accadono entrambe le cose?
Winther (2001) indaga su cellule e organismi come parti di un complesso da esplorare secondo due prospettive: il punto di vista dell'integrazione (parti di organismi sono moduli di livello intermedio coinvolti principalmente in processi meccanicistici) che abbraccia evoluzione e sviluppo; il punto di vista della competizione (parti come moduli che interagiscono impegnati principalmente in processi selettivi).

[*] Modularity: Understanding the Development and Evolution of Natural Complex Systems, W. Callebaut, D. Rasskin-Gutman, The MIT Press, Cambridge, MA, Cap. 3, 2005.
J.B. Mouret, S. Doncieux "Evolving modular neural-networks throungh exaptation" in Eleventh conference on Congress on Evolutionary Computation, 2009.
R. Calabretta "Connessionismo evolutivo e origine della modularità" in Scienze della mente, A. Borghi, T. Iacchini, Il Mulino, 2002.

2.1

Meccanismi di evoluzione dell'origine dei moduli

La selezione naturale agisce sul fenotipo per produrre moduli evolutivi. Un modello evolutivo è un insieme di caratteristiche fenotipiche altamente integrate da effetti pleiotropici (uno stesso gene controlla più di un carattere all'interno dello stesso fenotipo) di alcuni geni sottostanti e isolate da altri insiemi di geni dalla scarsità di tali effetti (Wagner & Altenberg, 1996). Un modello evolutivo che voglia spiegare l'origine dei moduli deve spiegare come la selezione naturale possa produrre questa distribuzione di effetti genetici.
Sono stati proposti diversi modelli teorici ma in tutti tranne uno, la modularità è direttamente o indirettamente connessa a qualche vantaggio selettivo (Calabretta e al., 2000; Force e al., 2004).
In uno studio sull'evoluzione della modularità funzionale (Calabretta, 2000), è stato scoperto un meccanismo che non può essere classificato come selezione diretta o indiretta della modularità in sé ma emerge da una specializzazione di moduli strutturali duplicati senza alcun beneficio intrinseco in termini di prestazioni o evoluzione. Calabretta ha utilizzato un modello con un algoritmo genetico che evolve architettura e pesi di una popolazione di reti neurali che controllano il comportamento di un robot mobile
Due comportamenti devono esser rappresentati da sottostrati neurali differenti, o deve esserci modularità funzionale?
In uno studio di Nolfi e al. (1997), la modularità funzionale non è necessaria per risolvere un problema di adattamento e l'algoritmo genetico risolve il problema senza che i comportamenti siano rappresentati da diversi elementi neuronali. Calabretta poi ha modificato il suo set simulativo, dove oltre alla mutazione dei pesi è stata permessa anche la duplicazione di elementi neurali. Per cui la modularità funzionale non comporta alcun beneficio intrinseco della capacità di adattamento.
Quale meccanismo dà origine a modularità funzionale?
Allora vi è la duplicazione di un'unità neuronale. Nel modello, tale passaggio non dà vantaggi da un punto di vista evolutivo in termini di fitness poiché le unità sono identiche. Successivamente si acquisisce un cambiamento nella regolazione delle unità duplicate che modificano il contesto ambientale. Segue poi un accumulo di mutazioni che permettono l'adattamento dell'unità di controllo al contesto funzionale in cui essa viene utilizzata più di frequente portando a un co-adattamento della parte regolatoria e funzionale della rete bloccando il sistema in uno stato specializzato funzionalmente. Dal punto di vista della genetica della popolazione, l'evoluzione della specializzazione funzionale è causata dalle interazioni fra geni che influenzano la particolare circostanza dove l'unità è attiva e i geni che controllano l'output prodotto. Esiste un meccanismo irreversibile fra mutazioni che influenzano il contesto ambientale e mutazioni che portano alla specializzazione dell'output al contesto ambientale in cui viene usato più di frequente.
In biologia evoluzionistica, sono stati identificati possibili meccanismi.

2.2

Selezione diretta per modularità

La selezione naturale causa modularità ma deve esserci una connessione fra vantaggio selettivo e modularità per cui uno degli effetti più rilevati della modularità è il suo potenziale impatto sull'aumento di possibilità di evoluzione (Altenberg, 1995; Galis, 1999, 2001; Gerhart e Kirschner, 1997; Holland, 1992; Liem, 1973; Riedl, 1978; Vermeij, 1970; Wagner e Altenberg, 1996). Quindi si pensa che evolva come risultato della selezione per evolvibilità (Gerhart e Kirschner, 1997; Riedl, 1978), è la prima possibilità. L'altra possibilità è che la modularità sia il risultato di mutazioni che spezzano vincoli di sviluppo causati da *linkage* di non adattamento fra caratteri (Leroi, 2000).

2.2.1

Selezione per evolvibilità

Il problema di come la modularità può essere spiegata come adattamento per evolvibilità è stato discusso nel contesto di come la selezione per evolvibilità può essere un fattore dell'evoluzione dell'architettura genetica. Il meccanismo è un semplice processo di selezione Darwiniano basato sulla differenza di fitness causata da differenze nei rate di adattamento fra cloni (Wagner, 1981). Funziona bene solo se non c'è ricombinazione.
Risultati di uno studio sull'evoluzione di effetti pleiotropici (Mezey, 2000) considera due caratteri, uno sotto selezione direzionale e l'altro sotto selezione stabilizzante. La selezione naturale agisce per cambiare un carattere ma molti altri dello stesso organismo rimangono sotto selezione stabilizzante. Gli effetti pleiotropici diminuiscono il rate di evoluzione del carattere sotto selezione direzionale (Baatz & Wagner, 1997). Gli effetti pleiotropici diminuiscono l'evolvibilità.
Il problema è se la selezione naturale potrebbe fissare un modificatore che sopprime il pleiotropico e poi aumentare l'evolvibilità? (Fig. 6). Usando un modello individuale per stimare i coefficienti di selezione del modificatore e misurando il tempo da fissazione si arriva a una rapida e forte selezione per il modificatore. Il coefficiente da solo non fornisce informazioni se sono distribuiti con selezione per evolvibilità. La fitness non buona dei genotipi con diversi modificatori è influenzata da almeno due fattori: la quantità di variazione nel carattere sotto selezione stabilizzante e la posizione dei genotipi dietro selezione direzionale (Fig. 7). Solo il secondo fattore può essere chiamato selezione per evolvibilità e deriva da rate di adattamento differenziali.

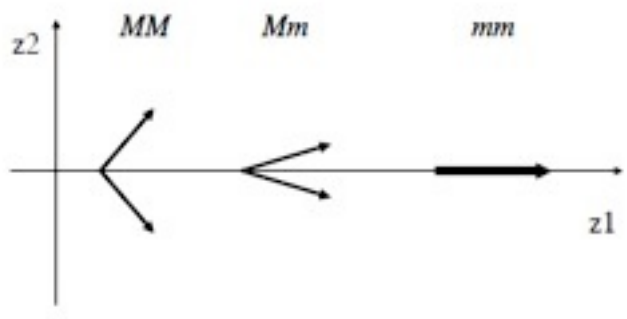


Fig. 6. Modello modificatore dove il genotipo alla posizione modificatore determina la dimensione degli effetti pleiotropici fra due caratteri. *MM* indica che gli effetti sui due caratteri della mutazione sono della stessa importanza; *Mm:* gli effetti su z2 sono più piccoli di quelli su z1; *mm*: mutazioni che non hanno effetto su z2. Il modificatore *m* sopprime gli effetti pleiotropici sul carattere sotto selezione stabilizzante. La sua selezione aumenta l'evolvibilità.

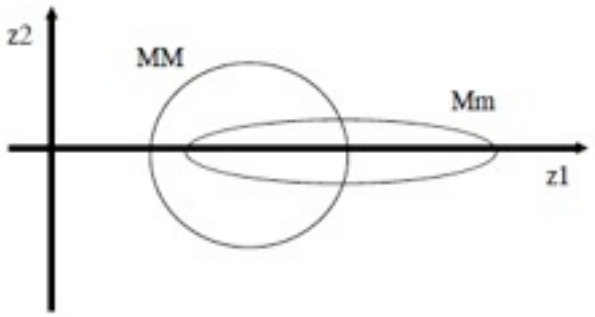


Fig. 7. Confronto distribuzione di valori di due classi di genotipi. MM hanno effetti mutazionali uguali sui due caratteri con distribuzione circolare. Mm ha effetti più piccoli su z2 con distribuzione più estesa in avanti sull'asse di z1.
Se il primo carattere è sotto selezione direzionale per valori più grandi e il secondo è sotto selezione direzionale, la fitness non buona è influenzata da due fattori. Il primo è la posizione della distribuzione in avanti sull'asse z1. Più la distribuzione è a destra, più alta è la fitness non buona. Il secondo è la quantità di variazione in z2, quindi z2 è sotto selezione stabilizzante, la fitness è più alta, la più piccola per z2. In questo caso, Mm ha fitness più alta. Solo il componente di questa fitness vantaggiosa dovuta alla posizione dietro l'asse di z1 può esser detto: selezione per Mm causato da selezione per evolvibilità.

Un altro studio sull'evoluzione per evolvibilità ha prodotto un risultato simile (Turney, 2000). Il modello ha considerato mutazioni che hanno aumentato la dimensione del fenotipo e il numero di gradi di libertà di variazione dell'adattamento. E' stato dimostrato che l'evolvibilità aumenta durante la simulazione. Il meccanismo evolutivo era un vantaggio selettivo diretto alle mutazioni che hanno aumentato la versatilità evolutiva (tali mutazioni prima inaccessibili).
Perciò l'evolvibilità può evolvere e migliorare ugualmente ma da sola non è il *target* della selezione. L'evoluzione della modularità è improbabile sia il risultato della selezione diretta a favore dell'evolvibilità. Questi risultati hanno suggeriscono due possibili meccanismi dell'origine dei moduli:
1) la mappa genotipo-fenotipo ha impatto diretto sulla fitness non buona in particolare se la popolazione è lontana dall'equilibrio (Rice, 1990). Quindi questa modularità risulta dal fatto che effetti pleiotropici possano diminuire la fitness non buona di una popolazione se questa sperimenta selezione direzionale;
2) la possibilità che mutazioni che producano modularità possano spezzare vincoli genetici sull'adattamento e poi successivamente potranno essere selezionate purché esse producano vantaggiosi fenotipi accessibili.

2.2.2

Selezione diretta su effetti pleiotropici

Basata sui risultati riportati sopra, si è provato a evolvere la modularità in un modello quantitativo genetico da combinazione di selezione direzionale su un carattere e selezione stabilizzante su un'altro carattere può portare a selezione contro effetti pleiotropici (Fig. 8). Su tale base si è evoluta la modularità alternando selezione direzionale fra due caratteri. La selezione direzionale su un singolo carattere seleziona contro gli effetti pleiotropici su altri caratteri. Se due caratteri non sperimentano selezione direzionale simultaneamente, può nascere un'architettura genetica modulare per i due caratteri (cioè un insieme di geni con la maggior parte degli effetti concentrati su un carattere e un'altro insieme di geni con la maggior parte dei propri effetti sull'altro carattere). I risultati hanno dimostrato che alternando la selezione non si arriva a una separazione di geni nei due insiemi di caratteri specifici, uno con effetti su un carattere e l'altro con effetti sull'altro carattere.

La distribuzione degli effetti dei geni non si stabilizza su un pattern modulare anzi ogni episodio di selezione direzionale tende a reclutare geni nel carattere selezionato (Mezey, 2000). Concludendo che alternando solo la selezione non si arriva all'origine della modularità evoluitva.

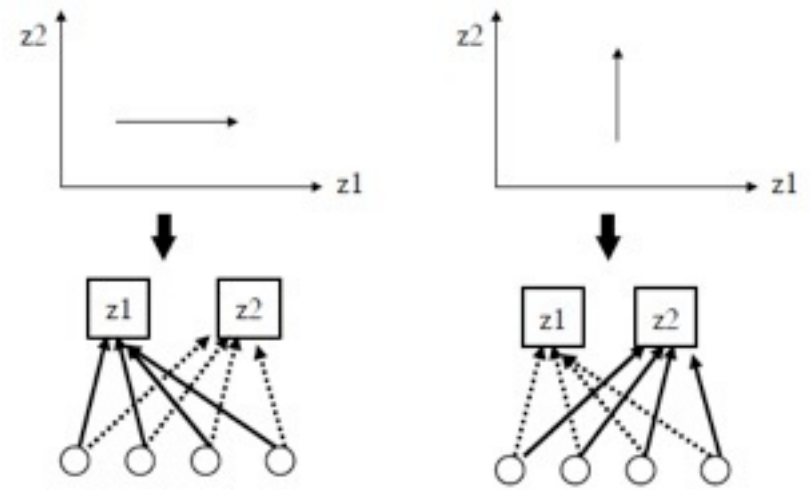


Fig. 8. Effetto della selezione direzionale su un carattere dell'architettura genetica di un fenotipo con 2 caratteri. Quando c'è selezione direzionale su ciascun carattere, tutti i geni aumentano il proprio contributo a quel carattere. Con stessa selezione direzionale ma alternando i due caratteri, non si arriva a segregazione di geni nel pattern modulare.

2.2.3

Modularità come fuga da vincoli di adattamento

La seconda alternativa menzionata sopra è che la modularità potrebbe derivare da mutazioni che superano i vincoli della capacità di adattamento fra caratteri. Quest'idea è legata al fatto che disaccoppiamento strutturale e funzionale può facilitare l'adattamento (Galis, 2001; Liem, 1973) ed è stato proposto come un meccanismo per l'origine della modularità da Leroi (2000).

2.3

Selezione costruttiva

È il modello più vecchio, proposto da Altenberg nel 1994 e più utilizzato nelle simulazioni. E' basato sul presupposto che geni con pochi effetti pleiotropici hanno maggiore probabilità di stabilire proprie copie duplicate di se stessi nel genoma. E' un modello basato sulla competizione intragenomica fra geni con diversi gradi di pleiotropia. Prevede l'evoluzione di più bassi gradi medi di pleiotropia. Il problema di questo modello è l'assunzione che il grado di pleiotropia è ereditabile fra copie di geni, in particolare se il gene acquisisce nuove funzioni. Infatti, risulta evidente per pleiotropia più bassa fra copie di geni duplicate ma può esser spiegato meglio dalla sottospecializzazione delle copie di gene dovuta a degenerazione e completamento fra elementi modulari migliorati (Force e al., 1999).

2.4

Stabilità fenotipica

E' la proprietà più selezionata e meglio "vista" da selezione naturale (Wagner e al., 1997). L'evoluzione della stabilità delle mutazioni e la modularità è una risposta correlata alla selezione sulla stabilità fenotipica.

2.5

Modularità che facilita adattamento fisiologico

Un altro modello simile è stata trovato da Calabretta e al. in un modello che simula l'evoluzione di una rete neurale artificiale dedicata a due task funzionali indipendenti, i cosiddetti task "What-Where" (Di Ferdinando, Calabretta e Parisi, 2001). La rete aspetta di produrre due tipi di uscite: una che indica la posizione di un oggetto e l'altra la sua identità (forma). Il modello porta all'evoluzione di un architettura neurale modulare con due componenti. L'architettura neurale, in particolare il fatto che i neuroni sono connessi ad ogni altro è determinata geneticamente ed è evoluta da mutazione e selezione. Dall'altra parte, la forza delle connessioni è determinata dall'algoritmo di apprendimento backpropagation (cioè acquisita da ogni individuo durante la sua ontogenesi). Tale modello, non porta all'evoluzione della modularità. La ragione dell'efficacia dell'algoritmo di apprendimento dipende dall'architettura neuronale. Solo un'architettura modulare fornisce le basi di un apprendimento di successo. Quindi la modularità determinata geneticamente, ha una fitness diretta vantaggiosa dovuta alla sua influenza sull'efficacia dell' apprendimento individuale.
In più, architetture neurali modulari sono anche modulari geneticamente rispetto a certe mutazioni e cioè in questo modello, la modularità genetica non evolve per le sue conseguenze variazionali (genetiche).
Senza architettura modulare la fitness del fenotipo è variabile, perché l'algoritmo di apprendimento non trova affidabili i pesi delle connessioni migliori (Di Ferdinando).

2.6

Modularità da "Frustrazione"

Uno studio di Rice (2000) ha scoperto un meccanismo inaspettato per l'origine dei moduli. Rice ha trovato che correlazioni positive evolvono se gli effetti di due caratteri sulla fitness sono sinergici, (cioè se l'aumento di un valore di un carattere aumenta la selezione direzionale sull'altro carattere). Dall'altra parte, l'evoluzione di una correlazione negativa è prevista se i caratteri sono antagonisti per quanto riguarda la fitness. Considerando più di due caratteri con coppie di interazioni antagoniste sulla fitness, accade qualcosa di inaspettato. E' impossibile avere correlazioni negative fra tre o più caratteri simultaneamente. L'evoluzione delle correlazioni negative per cui è detta "frustrata".
L'unica soluzione è che i caratteri evolvono da indipendenza variazionale e, quindi la modularità può derivare da interazione antagonista della fitness fra tre o più caratteri.

2.7

Modularità come *side effect* dinamico

In tutti i modelli descritti, la modularità è assunta essere collegata a qualche sorta di vantaggio selettivo.
Uno studio sull'evoluzione della modularità funzionale (Calabretta e al., 2000) ha portato alla scoperta di un meccanismo che non può essere classificato come selezione diretta o indiretta della modularità in sé. La modularità funzionale nasce da sottospecializzazione di moduli strutturali duplicati senza alcun beneficio intrinseco in termini di prestazioni o velocità di evoluzione. La modularità sorge interamente come effetto collaterale di dinamiche evolutive.

2.8

Evoluzione dell'evolvibilità e principi di congruenza

I risultati sulla possibilità di selezione diretta per evolvibilità sono mischiati. Molti dei modelli visti hanno dimostrato di creare modularità implicando qualche sorta di congruenza fra modularità e qualche proprietà direttamente selezionabile.
Il miglior esempio è fornito da Ancel & Fontana (2000) e riguarda la modularità nella struttura secondaria di RNA dove esiste correlazione fra gradi di modularità e stabilità fenotipica verso il rumore ambientale. Altri esempi come il modello sui task "What-Where" (Di Ferdinando e al., 2001) mira a una congruenza fra modularità genetica e fisiologica che porta a un vantaggio selettivo.
La selezione costruttiva assume congruenza fra pleiotropia variazionale e probabilità di fissazione di un gene duplicato, per cui l'evoluzione di effetti pleiotropici mira a una congruenza fra evolvibilità e fitness non buona in popolazioni non in equilibrio.
In natura non esiste meccanismo di specializzazione unico, ma una moltitudine che agiscono per produrre architetture genetiche modulari.

2.9

Moduli evolutivi: un'analisi concettuale

L'evoluzione della capacità di adattamento richiede l'esistenza di moduli evolutivi. Gli organismi devono essere scomponibili in *traits* che possono evolvere indipendentemente l'uno dall'altro.
Questo paragrafo si propone di mostrare lo scopo di distinguere fra un analisi concettuale della modularità evolutiva da ipotesi empiriche.
Secondo Wagner & Altenberg (1996) il concetto di "quasi indipendenza" di Lewontin (1978) e l'idea di reti geniche di Bonner (1988) sono concetti precursori del concetto di moduli evolutivi. Il punto è che entrambi pensano di aver dimostrato l'esistenza di tali moduli, indispensabili per l'evoluzione della capacità di adattamento e che tale evoluzione avviene. Questo viene classificato come un argomento "trascendentale" per l'esistenza di moduli evolutivi. Si distinguono tre ragioni per credere nell'esistenza di moduli evolutivi: l'argomento "trascendentale" di Lewontin e Bonner; l'interferenza "indiretta" sull'esistenza di certi moduli da dati filogenetici e l'osservazione "diretta" di moduli. I moduli evolutivi o "moduli selezione" sono tali in virtù di essere sia funzionali che modulari evolutivamente. Un modo di pensarli è come "unità" o "tipi naturali" selezionati dal processo di evoluzione attraverso la selezione naturale.

Il processo di evoluzione per selezione naturale si compone di due sottoprocessi: selezione fenotipica e risposta genetica alla selezione. A questi due sottoprocessi corrispondono due componenti di moduli evolutivi, funzione ecologica e modularità genetica/dello sviluppo. Un modulo evolutivo per essere tale deve avere una funzione unitaria: per "funzione unitaria" si intende la parte in questione nel suo complesso o una funzione come unità. Può averne più di una; le funzioni hanno struttura gerarchica.Quindi un modulo evolutivo è qualche caratteristica di un organismo con funzione ecologica e architettura genetica/di sviluppo che permette di evolvere in maniera "quasi-indipendente" da altre caratteristiche. Un modulo evolutivo deve essere funzionale e di sviluppo modulare.

2.10

Modello di modularità Wagner-Altenberg

Wagner e Altenberg hanno presentato un'immagine di moduli evolutivi molto dettagliata il cui modello è rappresentato in Fig. 9:

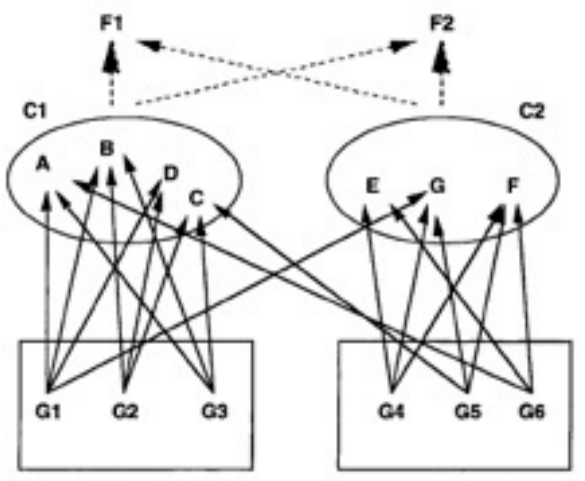


Fig. 9. Modello di modularità Wagner-Altenberg

Nella parte bassa ci sono due gruppi di geni: {G1, G2, G3}, {G4, G5, G6}. Ciò che rende ciascuno un insieme o nella terminologia di Bonner una "rete genica" sono le connessioni pleiotropiche in ogni insieme e scarsamente fra di loro. Le reti geniche sono insiemi di caratteri fenotipici, complessi di caratteri: {A, B, D, C}, {E, G, F} unificati dagli effetti funzionali. Il complesso C1 realizza la funzione 1 con effetto minore sulla funzione 2, C2 realizza la funzione 2 con un piccolo effetto sulla funzione 1. L'idea che un modulo evolutivo debba avere una funzione ecologica unitaria è rappresentata dall'effetto primario di C1 su F1 e analogamente di C2 su F2. L'idea che un modulo evolutivo debba avere un'architettura di sviluppo/genetica che permetta di evolvere indipendentemente da altri moduli è rappresentata dal fatto che la maggior parte delle frecce da geni a caratteri sono all'interno di un modulo. Per cui il modello Wagner-Altenberg è visto già come ipotesi empirica sulla natura dei moduli evolutivi e non come analisi concettuale.

2.11

Dimensioni della modularità

Oltre a criteri d integrazione interna e indipendenza da altri moduli, esistono altri due: nel caso di moduli evolutivi, devono *persistere* come unità identificabili per lunghi intervalli di tempo, generazioni, ed essere più o meno identici; devono essere *ripetitivi*, riutilizzabili e soggetti a cambiamenti al proprio interno. A livello più generale la modularità si presenta in due modalità: "potrebbe essere una struttura primaria del modo in cui gli organismi sono costruiti riguardante principi organizzativi di sistemi di auto-organizzazione", oppure potrebbe essere una "proprietà evoluta" (Wagner, 1996). Se l'organizzazione modulare è il prodotto dell'evoluzione da selezione naturale, la forza in grado di spiegare adattamento allo standard Darwiniano, potrebbe essere sia la *parcellation* (eliminazione di effetti pleiotropici fra caratteri appartenenti a gruppi diversi) o l'i*ntegrazione* di caratteri indipendenti che svolgono un ruolo funzionale comune (Fig. 10). La frequenza di entrambi è una questione empirica (Wagner, 1996).

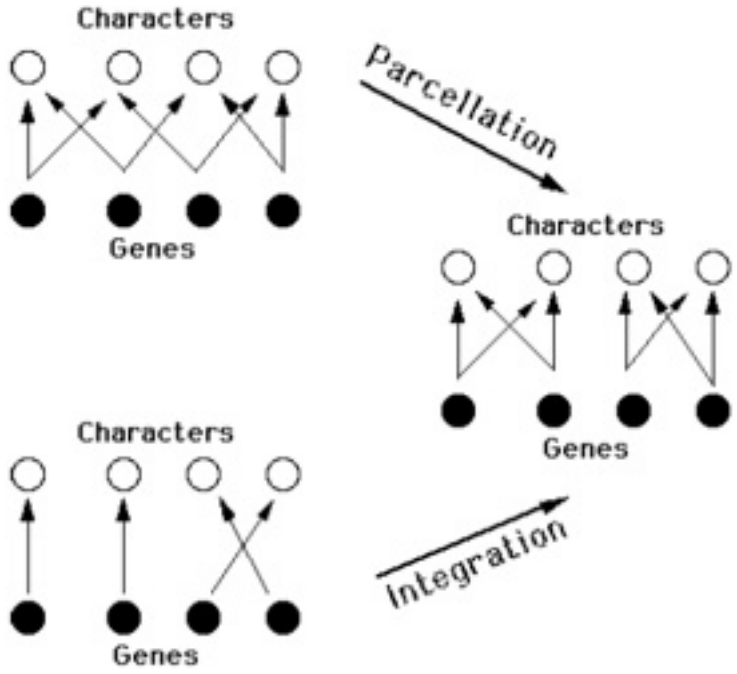


Fig. 10. Due modi di ottenere modularità

2.12

Struttura, Processo e Funzione

Della modularità occorre distinguere struttura, processo e funzione. Nel 1971, De Beer ha tracciato una linea netta fra struttura e funzione. La struttura considera la modularità come l'unica cosa adatta ad essere omologata, la funzione vista come un qualcosa che risulta essere omologa a un altra per quello che rappresenta e non per ciò che fa. Mentre la modularità di processo è indispensabile per dare significato alle funzioni modulari le quali rappresentano comportamenti e non strutture.
Moss (2001) fornisce una definizione di modulo come di "un unità, parte componente di un sistema più grande avente ancora una propria identità strutturale e/o funzionale".

2.13

Modularità attraverso *exaptation*

Mouret e al., [MDO09] hanno definito un problema di ottimizzazione multi obiettivo in cui ciascun obiettivo corrisponde a un sottotask e viene risolto con un algoritmo evolutivo multi obiettivo (MOEA) basato su Pareto. All'inizio del processo, gli individui hanno una fitness non minima solo per gli obiettivi semplici. Come sono migliorati tali individui per raggiungere prestazioni minime per i sottotask richiesti, per poi provare con quelli più complessi? Sono selezionati gli individui con fitness non minima su un sottotask non raggiungibile. Il processo evolutivo passa automaticamente ai task complessi, mantenendo una pressione selettiva sui task precedenti e che non dipende dal loro ordinamento. Inoltre, una parte della popolazione può essere selezionata rispetto a sottotask intermedi, un'altra può migliorare lungo un altro gradiente di selezione; il processo può poi esplorare simultaneamente diversi percorsi evolutivi. In questo caso, ogni gradiente di selezione si basa su soluzioni caratteristiche complete e non ha nessun impatto diretto sulle sottoparti, per cui differisce dall'evoluzione naturale.
Mouret & Doncieux assumono implicitamente che i sistemi modulari sono favoriti da processo evolutivo, non propongono qualsiasi collegamento esplicito fra selezione e moduli. Le reti monolitiche sono spesso più efficienti e più facili da ottenere rispetto alle reti modulari.

2.13.1

Mapping modulare genotipo-fenotipo

Per evolvere sistemi complessi è importante l'esistenza di moduli nel genotipo e nel fenotipo e come si collegano gli uni agli altri (*mapping*). Se il numero di archi fra gruppi è più piccolo del previsto, nella maggioranza dei casi è inferiore al numero di archi di una rete casuale con stesse dimensioni, tali sottografi costituiscono moduli di fenotipi. I moduli di genotipi sono più semplici, in quanto si devono controllare solo gli effetti degli operatori genetici su sottoparti del genotipo.
Il mapping è modulare se i moduli dei genotipi sono sviluppati nei moduli dei fenotipi. I moduli emergono dalla topologia della rete per cui un piccolo cambiamento, come l'aggiunta di una connessione indotta da mutazione, può modificare molto la suddivisione ottimale. Ciò rende difficile definire una mappa genotipo-fenotipo modulare: dipende dalla codifica della rete e potrebbe essere difficile garantire che un modulo di genotipo porti a un modulo di fenotipo significativo.

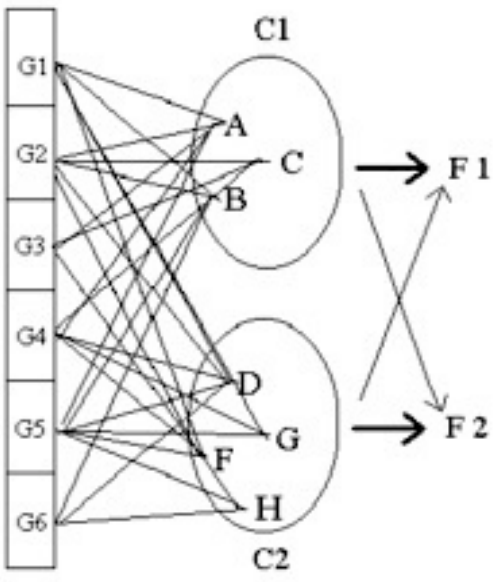


Fig. 14. Esempio di mapping genotipo-fenotipo non modulare.

Nel contesto biologico, come già esplicitato, Wagner e al., [WMC01] spiegano che solo due processi possono portare ad una mappa modulare genotipo-fenotipo: la parcellizzazione e l'integrazione.
Come questi due concetti, potrebbero essere trasposti alle reti neurali? Rimuovere gli effetti pleiotropici può essere interpretato come la possibilità di rendere moduli di fenotipi più resistenti alle mutazioni. La parcellizzazione può essere l'isolamento di una parte del genoma codificante per un modulo, cosi un modulo di fenotipo rimarrebbe stabile durante l'evoluzione. L'integrazione come la sostituzione di un modulo con una copia di un altro fornendo un meccanismo di ripetizione.
Sono possibili due tipi di mutazione:

- mutazione strutturale: aggiunta o rimozione di un neurone o connessione;
- mutazione parametrica: cambiamento del peso sinaptico scelto a caso o una polarizzazione neuronale.

Per aggiungere integrazione e parcellizzazione come operatori di mutazione, si definisce la rete codificata come una codifica diretta e una lista di sottoreti, ciascuna associata a collegamenti verso la rete principale (Fig. 11).

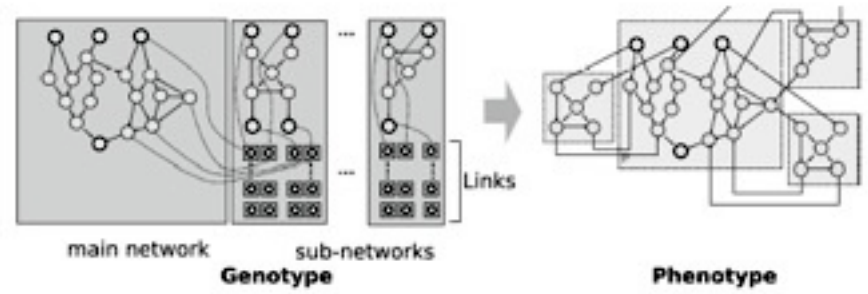


Fig. 11. Un genotipo è costituito da una rete principale e una o più sottoreti. Ogni sottorete è associata ad una lista di link che collegano le sottoreti alla rete principale. Se esistono più liste di link per una stessa sottorete, l'ultima è aggiunta più volte alla rete principale. A destra, la rete è un esempio di fenotipo che potrebbe esser codificato con il genotipo a sinistra.

Prima di applicare ciascun operatore, la rete principale viene suddivisa in moduli e identificati input e output. Gli input di un modulo sono entrambi una connessione fra neurone di un modulo esterno e quello di un modulo interno, oppure un input della rete principale. Gli output sono definiti in maniera simile.
La parcellizzazione si costituisce di tre fasi (Fig. 12a):

- rimozione un modulo dalla rete principale;
- aggiunta di una lista di sottoreti;
- aggiunta di link alla lista dei collegamenti delle sottoreti.

Il modulo rimosso può essere scelto a caso oppure in maniera deterministica come il miglior modulo in merito a uno dei sottotask. Anche l'integrazione si costituisce di tre fasi (Fig. 12b):

- selezione casuale di un modulo m_1 dalla lista delle sottoreti;
- selezione casuale di un modulo m_2 della rete principale con stesso numero di input e output come m_1;
- rimozione del modulo m_2 dalla rete principale;
- aggiunta di link alla lista di collegamenti della sottorete.

Per controbilanciare gli effetti dei due operatori, viene aggiunto un operatore chiamato "differenziazione" il quale inserisce nuovamente una sottorete nella rete principale e rimuove i collegamenti corrispondenti nella lista (Fig. 12c). La differenziazione annulla parcellizzazione e integrazione e può essere usata per creare una variante di un modulo ripetuto.

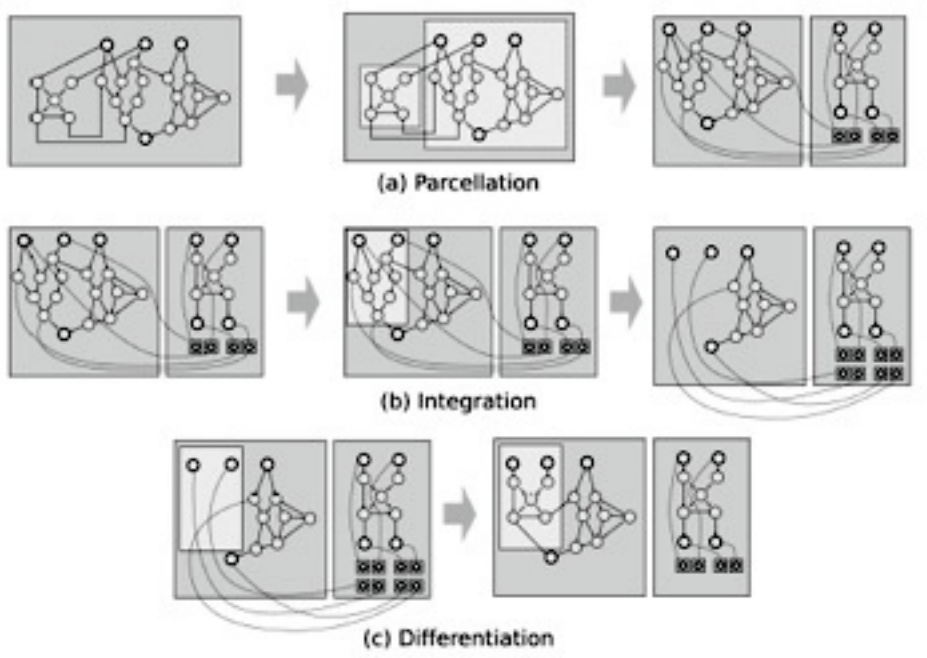


Fig. 12. Tre operatori di mutazione. **(a)** Parcellizzazione: isolamento di una sottorete. **(b)** Integrazione: sostituzione di una sottorete da un collegamento a quella *parcellated* (consente la ripetizione di un modulo). **(c)** Differenziazione: inserire una sottorete all'inizio della rete principale.

In Fig. 13 viene mostrato un semplice operatore crossover implementato scambiando due moduli *parcellated*.

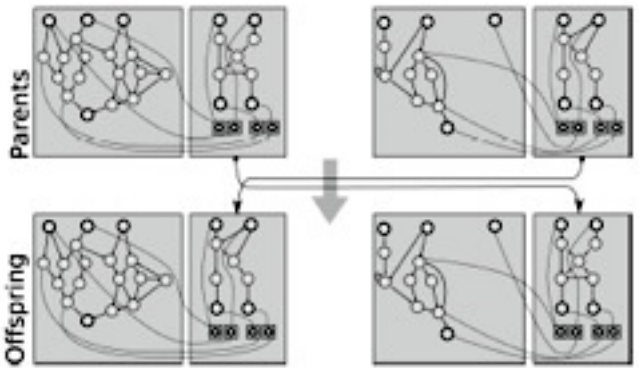


Fig. 13. Se entrambi i genitori hanno un modulo *parcellated*, viene scambiato per creare i figli. Altrimenti, ciascun figlio è una copia di uno dei genitori (sono poi applicati operatori di mutazione).

2.13.2

Gradienti di selezione

E' possibile allineare i moduli di fenotipi e gradienti di selezione, vale a dire che ogni modulo di fenotipo viene collegato a una funzione di fitness, definendo prima le funzioni di fitness capaci di valutare le sottoreti per sottofunzioni potenzialmente utili. Per ogni individuo, la rete principale è suddivisa in moduli. Ciascun modulo e ciascun modulo *parcellated* sono poi valutati a seconda dell'obiettivo. Viene usato il miglior valore ottenuto da un modulo per ogni obiettivo, come la fitness dei singoli rispetto a tale obiettivo. Ogni funzione di fitness costituisce un obiettivo del problema multi obiettivo. Con un MOEA basato su Pareto si hanno diverse conseguenze: un individuo con comportamento complessivo inefficiente sarà considerato Pareto-ottimale se contiene almeno un buon modulo per un sottotask. Se il task principale richiede un modulo efficiente per costruire più soluzioni complesse, sarà migliorato gradualmente grazie alla selezione introdotta dal corrispondente obiettivo. Successivamente, questo modulo potrà essere co-optato in qualsiasi momento. Lo stesso modulo potrà essere ripetuto o propagato nella popolazione mediante crossover. Altra importante conseguenza è che risolvere ogni sottotask non è obbligatorio.
Se il processo evolutivo trova le soluzioni migliori senza usare moduli, questi individui domineranno l'obiettivo principale e sarà conseguentemente selezionato.
Diverse ipotesi sull'utilità di ciascun sottotask possono essere studiate in una sola volta: alcuni individui saranno buoni a un sottotask, mentre altri conterranno moduli utili per altri sottotask. Il processo evolutivo li modificherà opportunamente, senza conoscenza a priori del loro ordine o utilità.

2.14

Tipi di modularità

Si distinguono diversi tipi di modularità, in particolare la modularità biologica si suddivide in tre aspetti riguardanti lo sviluppo, la morfologia e l'evoluzione.

2.14.1

Modularità biologica: sviluppo

Nella figura che segue, è riportato un esempio di mapping non modulare.

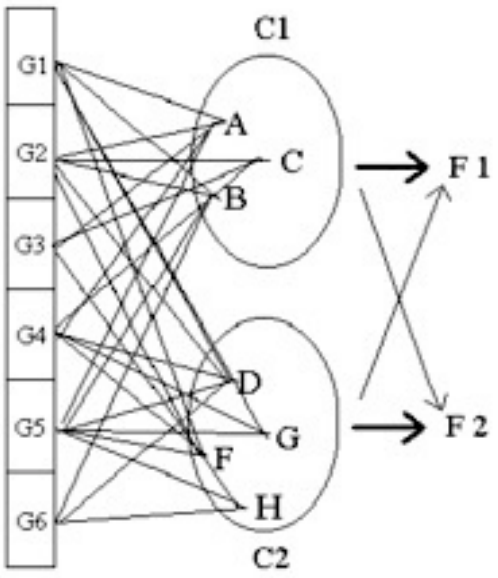


Fig. 14. Esempio di mapping genotipo-fenotipo non modulare.

Alcuni adattamenti possono avere una rappresentazione genetica modulare intrinseca perché sono semplici e includono un azione diretta del gene. La morfogenesi presenta maggiore cambiamento nel produrre una rappresentazione modulare, dal momento che un sistema dinamico emerge da interazioni di più geni e strutture.
L'idea di sviluppo come un organizzazione in processi semiautonomi non è recente. L'evoluzione di adattamenti complessi richiede corrispondenza fra rapporti funzionali di caratteri di fenotipi e rappresentazione genetica.
Riedl (1978) prevedeva una selezione che tendesse a favorire mappe genotipo-fenotipo che imitano l'organizzazione funzionale di caratteri. Per "imitazione" intendeva: complessi di caratteri funzionali collegati "codificati" come caratteri integrati ma indipendentemente da complessi di caratteri distinti. L'esistenza di unità semiautonome di fenotipo potrebbe esser importante nella connessione con riproduzione sessuale (Stearns, 1993). Essa riordina la variazione genetica in ogni generazione creando cosi il problema di complessi di caratteri funzionali connessi si consente una ricombinazione di unità integrate piuttosto che una variazione "casuale".

2.14.2

Modularità morfologica

La modularità morfologica è vista come la preservazione dell'integrità funzionale di una parte e non della funzione obiettivo. Una parte vista come modulo, potrebbe esser chiamata "attività" di un organismo, per esempio, nella sua fisiologia o nel suo comportamento anziché nel suo sviluppo.
La modularità morfologica è vista come un aspetto della modularità biologica riguardante lo sviluppo.

2.14.3

Modularità evolutiva

Wagner & Altenberg (1996) definirono moduli evolutivi da una variazione indipendente e dall'integrazione fra parti, o dalla variazione interspecifica o mutazionale. Analogamente, un modulo di selezione può essere definito come un insieme di geni, prodotti e interazioni, come carattere complesso risultante ed effetto funzionale del complesso.
Brandon (1999) dà una definizione di modulo evolutivo come quello dell'evoluzione per selezione naturale "il modulo evolutivo individua, seleziona e trasforma". Tali moduli rappresentano unità di evoluzione da selezione naturale.

2.14.4

Modularità neurale e cognitiva

La concezione modulare del cervello umano fu quella di un "modulo magico" neurologico in grado di spiegare il linguaggio umano e simbolico. Secondo Chomsky, le proprietà del linguaggio umano richiedevano un dispositivo celebrale incorporato per la sua generazione. Contrariamente Deacon, offre una visione alternativa nella quale linguaggio e cervello evolvono insieme identificando vari moduli del cervello a vari livelli di granularità.
Fodor (1983) sosteneva l'esistenza di due classi principali di entità cognitive nel cervello: moduli mentali di dominio-specifico che includono l'inconscio, computazioni visive; altri sistemi di input come sistemi di output che rappresentano per comportamento un dominio-generale.
Riferendosi a Chomsky, Fodor pensava che il linguaggio fosse uno dei moduli della mente. L'essenza della modularità Fodoriana è "l'incapsulamento dell'informazione": qualche informazione all'esterno del modulo non è accessibile dall'interno. Restringendo il flusso di informazioni si arriva a dire che i moduli sono "*mandatory*", non si può controllare se un modulo si applica a un dato input; sono solitamente veloci rispetto a processi non modulari; sono "computazionalmente superficiali" in quanto forniscono solo una caratterizzazione preliminare dell'output; meccanismi modulari associati a un'architettura neurale fissa e come conseguenza possiedono modelli di composizione caratteristici.
Sulla base della selezione naturale è interessante notare perché uno degli obiettivi primari è spiegare l'organizzazione funzionale degli organismi. La selezione deve produrre prodotti altamente specializzati per adattarsi a specifiche condizioni ambientali che causano problemi di adattamento.
Di conseguenza, il cervello dovrebbe essere costituito di una serie di moduli dedicati, ciascuno predisposto ad affrontare un problema di adattamento. Una capacità modulare può essere acquisita in quattro modi distinti:
1) la capacità innata è fissa ma deve essere attivata opportunamente al fine di svilupparsi appieno;

2) è determinata impostando parametri ambientali;
3) ha una base innata che poi rimane fissa per lo sviluppo di moduli interni, usando solo informazioni "consentite" oltre il confine informativo del modulo;
4) la capacità può essere "cognitivamente penetrabile" e appresa per induzione.

2.15

Connessionismo evolutivo

Problemi ancora non risolti riguardano i vantaggi di un'organizzazione modulare rispetto a una non modulare, meccanismi che favoriscono l'emergere evolutivo della modularità e quali, al contrario ne ritardano o ne impediscono l'insorgenza.
Tali meccanismi sono uguali per tutti i moduli, oppure esiste uno dedicato per ogni parte?
I moduli evolutivi e di sviluppo sono o no la stessa cosa? Quale relazione c'è fra i due tipi di moduli?
La modularità viene affrontata nelle scienze cognitive in cui la mente è una collezione di moduli specializzati in diverse abilità e aree di attività. Sarebbe innata, ossia codificata nel genoma e presente dalla nascita.
Gli empiristi sottolineano il ruolo del processo di sviluppo della mente e quindi una posizione anti-innatista. In questo ambito il connessionismo (Rumelhart & McClelland, 1986) usa reti neurali come modello del cervello per dimostrare le alte capacità di apprendimento di reti modulari/non-modulari che partono da una condizione di mancanza totale di competenza nel compito testato.

2.16

Modularità della mente

Le scienze cognitive sono due: cognitivismo e connessionismo.
Cognitivismo: paradigma teorico, nato negli Stati Uniti verso la fine degli anni 50, basato su analogia tra mente e computer. La mente come manipolazione di simboli.
Connessionismo: emerso negli ultimi 20 anni, rifiuta analogia mente/computer e interpreta il comportamento e le abilità cognitive con modelli teorici ispirati alla struttura fisica e al funzionamento del sistema nervoso. Tali modelli sono le reti neurali. Per il connessionismo, la mente non è il risultato globale di tante interazioni che avvengono nella rete di neuroni ma consiste solo di processi quantitativi, in cui cause fisico-chimiche producono effetti fisico-chimici.
Il cognitivismo è fortemente modularista: la mente sarebbe formata da moduli distinti specializzati per elaborare tipi di informazione diversa. Tali modelli sono i cosiddetti "diagrammi scatole e frecce": ogni scatola rappresenta un modulo con una specifica funzione; le frecce collegano le scatole e indicano che l'informazione elaborata da un modulo viene poi trasferita a un altro per essere elaborata ancora.
Il connessionismo tende a essere anti-modularista. Nelle reti, l'informazione è rappresentata da pattern di attivazione, distribuiti in grandi gruppi di neuroni e il modo di funzionare consiste nella trasformazione dei pattern di attivazione in altri pattern di attivazione, ciò si realizza lungo le connessioni dei diversi neuroni.
Le reti usate come modelli del cervello sono in gran parte non modulari se si effettua la distinzione in unità di input, unità nascoste e unità di output (Di Ferdinando, 2001).
Il punto di contrasto più importante fra le due scienze cognitive riguarda l'innatismo e l'anti-innatismo.
I cognitivisti sono innatisti e le informazioni sui moduli sono specificate nel genoma degli individui. La struttura modulare della mente deve essere il risultato di pressioni selettive.

Cosmides & Tooby (1994) sostengono una forma di adattamento in cui i moduli non sono codificati nei geni ma il risultato di processi di adattamento. Forma non condivisa da tutti i cognitivisti.
Gould (1997) invece, sostiene che ciò che viene ereditato geneticamente non è il risultato di pressioni selettive e non è necessariamente capace di adattamento, ma il risultato di fattori casuali che accompagnano un altro *trait* di adattamento, oppure potrebbe essere un *trait* che si è evoluto perchè svolgeva una funzione e poi riutilizzato per una nuova funzione.
Il contrasto fra Pinker (1999) e Fodor (1998) cognitivisti e innatisti, ha dimostrato come la capacità di adattamento dei tratti ereditari può dividere gli stessi cognitivisti. Pinker favorevole a una forte forma di modularismo innato, Fodor per una forma di modularismo non innato.
I connessionisti sono anti-innatisti, al contrario dei cognitivisti. Il connessionismo è solitamente associato a posizioni empiriste le quali considerano la mente come frutto dell'apprendimento e dell'esperienza in vita. La materia si complica se si considera anche lo sviluppo, il quale è il processo di formazione dell'organismo adulto (fenotipo) a partire dall'informazione genetica (genotipo). Il processo di sviluppo non è un processo istantaneo ma temporale, consiste di una successione temporale di forme fenotipiche che portano indirettamente a considerare l'importanza dell'apprendimento e dell'esperienza sul fenotipo.
I cognitivisti sminuiscono anche il ruolo dello sviluppo. Calabretta sostiene una forma di connessionismo, né anti-modularista né anti-innatista. Il connessionismo non è necessariamente anti-innatista.
Anche se molti pattern di reti usano algoritmi di apprendimento per trovare pesi adatti a svolgere un particolare *task*, il connessionismo non è incompatibile con il riconoscere che alcuni aspetti della rete non siano il risultato dell'apprendimento ma ereditati.
Dato che la maggior parte delle simulazioni connessionistiche parte da un'architettura fissa, si potrebbe ipotizzare che sia determinata geneticamente mentre il ruolo dell'apprendimento limitato a trovare i pesi adatti. L'architettura definisce come sono collegate fra loro le varie unità che la compongono.
Elman e al. (1996) sostengono reti connessionistiche capaci di andare oltre il cognitivismo, cosa può essere innato e appreso, mostrando come caratteristiche fenotipiche sono il risultato dell'interazione fra ciò che è innato e ciò che è appreso, distinguendo cosi tre cose che potrebbero essere innate in una rete: i pesi delle connessioni, i vincoli architetturali e i vincoli cronotopici (cioè determinano quando alcune cose devono succedere durante lo sviluppo). Per ipotizzare che qualcosa è innato, oltre a implementare a priori delle proprietà, bisogna simulare realmente il processo evolutivo.
Le simulazioni (Holland, 1992) usano algoritmi genetici per simulare processi evolutivi ed evolvere proprietà di reti neurali (Parisi e al.,1990; Calabretta e al., 1996). Simulano non una singola rete che apprende abilità in base alla propria esperienza individuale ma un'intera popolazione di reti che passa attraverso una successione di generazioni di individui, ciascuno con genotipo ereditato. Usare l'algoritmo genetico mostra come l'informazione codificata nei genotipi ereditati cambia nelle generazioni, poiché la riproduzione è selettiva e nuove varianti dei genotipi vengono aggiunti costantemente allla popolazione grazie a mutazioni genetiche e ricombinazione. A simulazione conclusa, i genotipi ereditati avranno codificato le proprietà desiderate rappresentanti vincoli innati allo sviluppo e comportamento. Questo tipo di connessionismo è chiamato evolutivo: come l'informazione ereditata geneticamente può emergere spontaneamente in popolazioni di reti anziché essere implementata arbitrariamente, esplora anche tutte le possibili interazioni fra evoluzione a livello di popolazione e apprendimento a livello del singolo che determina il fenotipo reale. Tale approccio consente di ricreare diversi scenari evolutivi ipotizzati alla base dell'evoluzione della modularità della mente; studiare il ruolo e l'interazione fra evoluzione, sviluppo e apprendimento; spiegare come alcune specie sono modulari e altre no; comprendere dinamiche di base della modularità del cervello.
Il modularismo cognitivista è diverso dal modularismo neurale. Per i cognitivisti, i moduli sono componenti di teorie in base ai quali fenomeni empirici sono interpretati e spiegati. Sono entità postulate più che osservate.

Lo stesso vale per la Psicologia Evolutiva che vede la mente come un insieme di moduli capaci di adattamento specializzati ed ereditati, basandosi su una nozione di modulo come entità teorica la cui esistenza è data da un'osservazione del comportamento umano.
Quindi le reti neurali sono allo stesso tempo sia modelli del cervello che della mente. Le reti neurali usate nella maggior parte delle simulazioni sono di tipo non-modulare, omogenee, costituite di unità con una struttura semplice e da un insieme di unità o da un modulo di input, uno di output e in mezzo quasi sempre un singolo modulo interno di unità nascoste.

COGNITIVISMO	Mente come manipolazione di simboli che avviene in un sistema analogo al computer	INNATISTA	MODULARISTA
CONNESSIONISMO	Mente come risultato globale delle moltissime interazioni fisico-chimiche che avvengono in una rete di neuroni	ANTI-INNATISTA	ANTI-MODULARISTA
CONNESSIONISMO EVOLUTIVO	Mente come risultato globale delle moltissime interazioni fisico-chimiche che avvengono in una rete di neuroni	INTERAZIONE TRA EVOLUZIONE E APPRENDIMENTO	MODULARISTA

Fig. 15. Tre opzioni per studiare comportamento e mente.

I moduli delle reti neurali sono più simili ai moduli del cervello che le "scatole" teoriche dei modelli "scatole e frecce" del cognitivismo. Un modulo in una rete neurale è un modulo fisico (simulato) e non un costrutto teorico postulato, costituito da alcune unità con un numero maggiore di connessioni interne rispetto al numero di connessioni esterne che collegano le unità del modulo alle unità esterne.
Da un punto di vista funzionale, un modulo neurale potrà essere l'attività correlata osservata di un dato sottogruppo di unità, senza separazione anatomica. Se la struttura modulare è implementata manualmente, si dovrà trarre ispirazione dalla struttura reale del cervello. Al contrario se è il risultato di un processo di evoluzione, sviluppo, e/o apprendimento, è importante capire se la struttura modulare emergente corrisponde alla reale modularità del cervello.
Il contrasto fra modelli a reti neurali e cognitivi, non riguarda la modularità in sé, ma la natura. Quali modelli teorici sono più adatti per spiegare il comportamento? Quelli basati su reti neurali ispirati al cervello. Il cervello è un sistema fisico con entità, processi fisici e ciò che avviene è la produzione di fenomeni fisico-chimici per cause fisico-chimiche. Quindi una rete neurale non può servirsi di una regola come spiegazione di alcun tipo di comportamento o di abilità cognitiva.

CAPITOLO 3

Evoluzione della modularità [*]

La capacità degli organismi naturali di evolvere è chiamata evolvibilità o possibilità di evoluzione e rappresenta un rapido adattamento a nuovi ambienti. L'evolvibilità è dovuta alla modularità che consiste in un'organizzazione funzionale con pochi collegamenti con sottounità.
Una rete è definita modulare se ha cluster di nodi densamente connessi fra di loro ma scarsamente connessi con nodi di altri cluster.
Come si evolve la modularità?

- mediante selezione indiretta dell'evolvibilità;
- mediante selezione diretta: consiste nel minimizzare il costo delle connessioni fra i nodi della rete causando la nascita di reti modulari.

La selezione per massimizzare le prestazioni e minimizzare i costi produce reti più modulari e capaci di evolvere di una selezione per sola prestazione.
Intuitivamente i sistemi modulari sembrano avere maggiore capacità di adattamento, per cui è più facile ricostruire una rete modulare in sottounità funzionali.
La possibilità di evoluzione fornisce un vantaggio selettivo soltanto a lungo termine, è migliore della selezione indiretta ma non basta a spiegare il livello di modularità presente nel mondo naturale [WPC07, WMC01]. La modularità è causata probabilmente da più forze che agiscono a vari livelli e in contesti diversi [WPC07].
Alcune ipotesi [KAL05, KNA07] sostengono che la modularità emerga per rapidi cambiamenti ambientali in sottoproblemi comuni ma diversi nel complesso, ambienti chiamati *modularly varying goals MVG*, in cui le reti evolvono in modularità ed evolvibilità. Mentre l'evoluzione in ambienti immutabili produce reti non modulari e più lente ad adattarsi al nuovo ambiente.
La forza che genera modularità di MVG in natura è legata alla frequenza con la quale gli ambienti cambiano. Non è possibile giustificare tanta modularità in MVG naturali poiché non si conosce come cambiano in modularità e, se cambiano ad una frequenza sufficientemente alta tale da rivestire un ruolo significativo.
Una teoria [ESW10] presuppone un ambiente in costante evoluzione, una selezione per evolvibilità e che la modularità nasca per modificare un sottocomponente senza influire gli altri.
Wagner, nelle sue ipotesi [WPC07] include la variazione di alcuni meccanismi, come la duplicazione dei geni che tende a generare strutture modulari, cosi la modularità evolve per rendere più robusti i fenotipi ai cambiamenti ambientali.

[*] J. Clune, J.B. Mouret, H. Lipson "The evolutionary origins of modularity" in Proceedings of The Royal Society: Biological Science, N.280, 2013.

Ipotesi [CAJ99], [STR05] suggerita ma non provata è che la modularità evolva come sottoprodotto della selezione per minimizzare i costi di connessione (Fig.16).

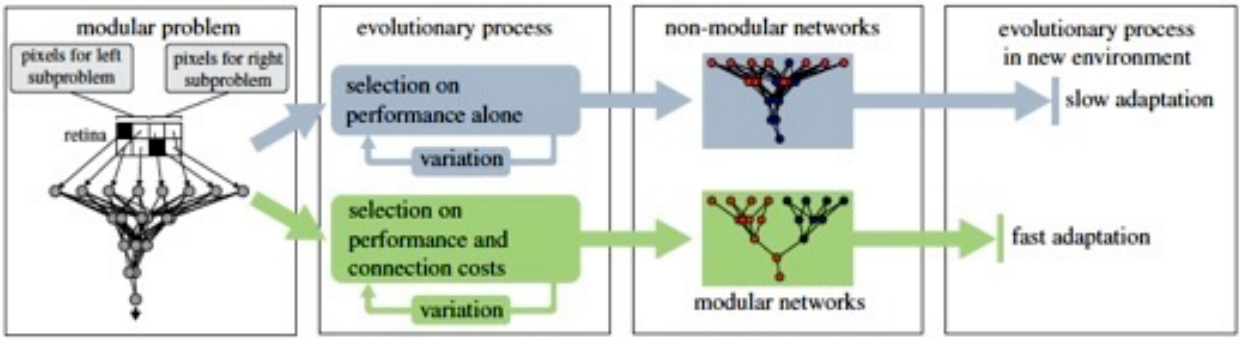


Fig. 16. Evoluzione di reti con selezione per sola prestazione produce reti non modulari lente ad adattarsi a nuovi ambienti. Aggiungendo una selezione che minimizza i costi si va verso l'evoluzione di reti modulari che si adattano rapidamente a nuovi ambienti.

Secondo un altro studio sulla non evoluzione [JJO92] che considera reti neurali simulate con uno specifico algoritmo di apprendimento in vita, questo sviluppa maggiore modularità invece di minimizzare la lunghezza delle connessioni e le prestazioni. Senza tali algoritmi, ambienti MVG o il cambiamento di operatori non portano alla creazione di moduli.

3.1

Modularità in sistemi computazionali

L'evoluzione di un sistema computazionale che ci interessa riguarda reti che risolvono Task di Pattern-Recognition e Task Logici Booleani.
L'input rileva l'ambiente e l'output determina le prestazioni sui problemi ambientali. Si confronta la fitness basata solo sulle prestazioni (PA) e la fitness basata su due obiettivi: massimizzare le prestazioni e minimizzare i costi di connessione (P&CC). Si utilizza un algoritmo di evoluzione multi obiettivo con un obiettivo PA o due obiettivi P&CC.
La selezione risulta più forte sulle prestazioni che sui costi. Le funzioni di costo di connessione sono due: una di default, la lunghezza al quadrato di tutti i collegamenti assumendo una posizione ottima dei nodi per fare in modo che tale somma sia minima; un altra come numero di collegamenti, la quale produce risultati qualitativamente simili ma rappresenta meglio reti biologiche senza connessioni con lunghezze diverse.

3.2

Casi di studio

Un caso di studio coinvolge una rete che riceve stimoli da otto input. E' pensata come una retina di otto pixel che riceve stimoli visivi. Sulla parte sinistra e destra della retina vengono visualizzati i pattern. Entrambe possono contenere un "oggetto" che rappresenta un pattern di interesse (Fig. 17a). Le reti evolvono rispondendo se l'oggetto è presente su entrambi i lati (ambiente L-AND-R) oppure solo su uno (ambiente L-OR-R).
Ogni rete vede iterativamente tutti i 256 possibili pattern di input e si hanno risposte vere (≥ 0) o false (< 0). La percentuale di risposte corrette rappresenta la prestazione e dipende dai neuroni collegati come forza e dal tipo di collegamento che può essere inibitorio o eccitatorio.
All'inizio di ogni esperimento, le reti sono generate a caso e le connessioni cambiate sempre a caso durante la replica.
La modularità della rete è valutata con una misura basata sull'approccio di Newman & Girvan. L'algoritmo di Newman & Girvan trova la suddivisione di nodi in moduli che massimizza Q. Questa misura prima divide le reti in moduli in maniera ottimale e poi misura la differenza fra numero di archi all'interno di ciascun modulo e numero previsto per reti casuali aventi uno stesso numero di archi.

$$Q = \sum_{s=1}^{K} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right]$$

K è il numero di moduli, L il numero di archi nella rete, l_s il numero di archi fra i nodi nel modulo s, d_s è la somma dei layer dei nodi nel modulo s.
La logica è: una buona suddivisione della rete in moduli deve comprendere molti archi all'interno del modulo e il minor numero possibile tra-modulo. Provando a minimizzare il numero di archi tra-modulo (o in maniera equivalente massimizzare quelli all'interno), la suddivisione ottimale consiste di un singolo modulo e non archi tra-modulo. L'equazione esposta sopra risponde a questa difficoltà imponendo $Q = 0$ se i nodi sono a caso nei moduli o se tutti i nodi sono in uno stesso modulo.
Si può normalizzare tale misura rispetto a reti randomizzate e si arriva a una misura Q_m normalizzata:

$$Q_m = (Q_{real} - Q_{rand}) / (Q_{max} - Q_{rand})$$

Q_{real} : valore Q della rete, Q_{rand} : valor medio Q delle reti randomizzate, Q_{max} : valore massimo possibile con stessa sequenza come grado vero e proprio della rete. Per calcolare Q_m, si trasforma la rete in un grafo non orientato ignorando archi direzionali e calcolando la sua Q_{real}. Per misurare Q_{rand}, si fanno due controlli, i quali forniscono risultati simili. Per il primo controllo, si usano reti randomizzate che preservano la sequenza della rete reale; per il secondo controllo, si calcola Q di reti codificate da genomi casuali, mappate su reti con stesso numero di nodi come nella rete reale.

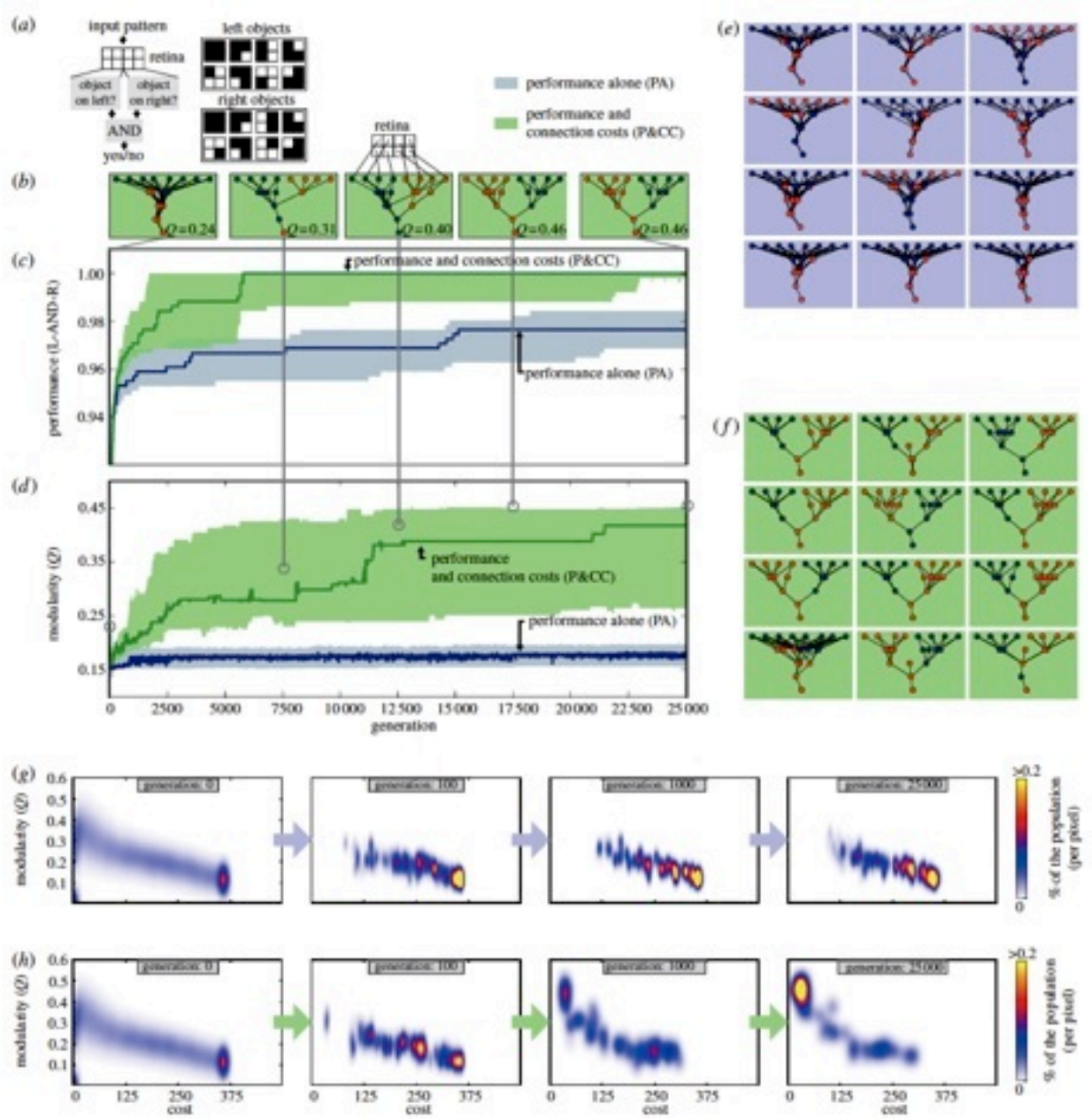


Fig. 17. Aggiungendo costi di connessione si producono reti modulari funzionali ad alte prestazioni. **(a)** Evoluzione di reti per riconoscere i pattern nella retina di 8 pixel. Problema scomponibile per modularità. Se un oggetto esiste su entrambi i lati, può essere determinato separatamente, combinando prima questa informazione per rispondere poi se esistono su entrambi i lati (AND). **(b)** Reti di un esempio di prova diventano più modulari evolvendo con una pressione (P&CC). **(c)** Prestazione media (+95%) che genera la rete con più prestazione ad ogni prova, risulta perfetta solo quando si minimizza (P&CC) **(d)** Modularità della rete più alta in P&CC rispetto a PA. **(e)** 12 reti PA con prestazione più alta, ciascuna da una prova separata. **(f)** 12 reti P&CC con prestazione più alta, funzionalmente modulari con moduli separati per i sottoproblemi sinistro/destro. I nodi colorati in base all'appartenenza a parti separate quando è possibile una divisione modulare della rete. **(g, h)** Costi e modularità di popolazioni PA e P&CC nelle generazioni da 50 prove.

Nell'esperimento esposto sopra, dopo 25.000 generazioni in ambiente immutabile L-AND-R, P&CC produce reti più modulari di PA. Le reti evolute che presentano modularità funzionale corrispondente alla scomposizione sinistra/destra del task, sono visualizzate con una suddivisione delle reti in due moduli, una suddivisione che massimizza Q e una per i nodi colorati per ogni partizione diversa. La scomposizione sinistra/destra è evidente in più prove P&CC e assente in prove PA (Fig. 17e, f).

La modularità funzionale è quantificata identificando se gli input a sinistra e destra sono in partizioni diverse. Nonostante l'ulteriore vincolo, reti P&CC presentano prestazioni più alte di reti PA (Fig. 17c). La prestazione media di P&CC esegue perfettamente a differenza di PA.
La prestazione P&CC è più alta perché le reti hanno meno nodi e connessioni, quindi meno parametri da ottimizzare.
Per capire meglio perché la presenza di un costo aumenti la prestazione e la modularità, si è cercato reti con prestazioni alte a tutte le combinazioni possibili di modularità e costo.
Per reti con alte prestazioni esiste una correlazione inversa fra costo e modularità: reti a più basso costo sono altamente modulari (Fig. 18). Ci sono anche molte reti non modulari ad alto costo, con alte prestazioni che spiegano perché la modularità non evolve per PA (Fig. 18).

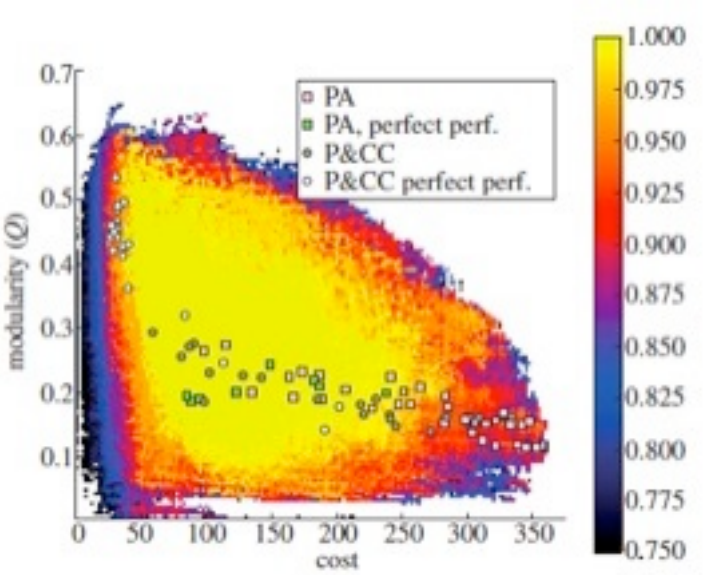


Fig. 18. Reti con prestazioni alte per combinazione di modularità e costo. I colori indicano la rete con prestazione alta trovata in quel punto della modularità rispetto al costo. Giallo: prestazione perfetta. Le reti con prestazione migliore alla fine di ciascuna delle 50 prove PA e P&CC sono sovrapposte. Reti con prestazione perfetta in tutto lo spazio spiegano perché la modularità non evolve con solo PA. Sotto una soglia di costo di circa 125 c'è correlazione inversa fra costo e modularità per reti con prestazione perfetta.

Confrontando nelle generazioni popolazioni PA e P&CC, si ha che un costo spinge fuori dall'area, popolazioni con alto costo, aree con bassa modularità dello spazio di ricerca nel basso costo e aree modulari (Fig. 17g, h). Senza la pressione di lasciare in alto costo regioni a bassa modularità, molte reti PA rimangono in aree che non contengono all'infinito soluzioni con più alte prestazioni (Fig. 18, quadratini verde scuro in basso a destra), ulteriore spiegazione del perché le P&CC hanno prestazione più alta. Reti P&CC evolvono di più rispetto a PA. Sono state eseguite 50 prove P&CC e 50 PA, ognuna con una rete con prestazione perfetta, poi trasferite in ambiente L-OR-R con stessi sottoproblemi ma in una combinazione diversa. La presenza (P&CC) o assenza (PA) di un costo è rimasto, anche dopo il cambiamento dell'ambiente.
Ripetendo ed evolvendo prima in ambiente L-OR-R e poi trasferendo in ambiente L-AND-R si ha in entrambi che reti P&CC hanno maggiore possibilità di evoluzione di PA richiedendo a meno generazioni di adattarsi al nuovo ambiente (Fig. 19a).

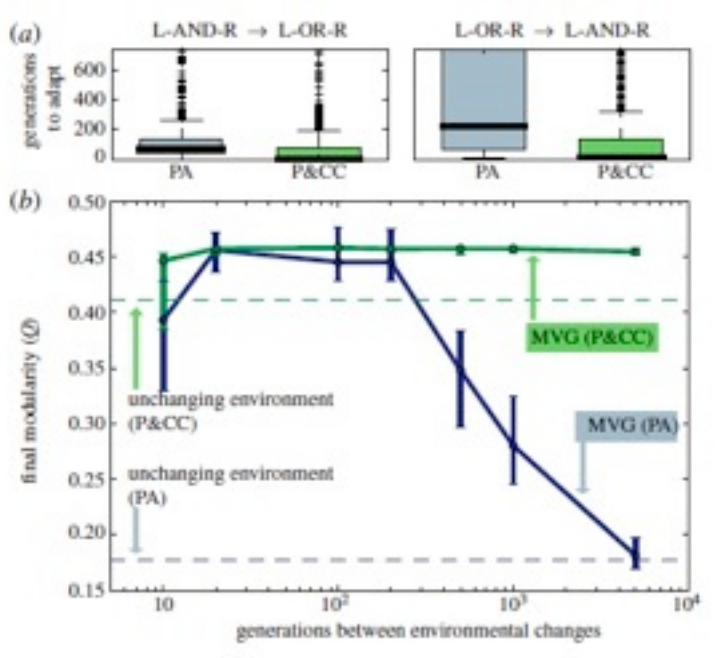


Fig. 19. Evoluzione con costi di connessione produce reti che evolvono di più. **(a)** Reti P&CC si adattano più velocemente a nuovi ambienti di reti PA. Evoluzione in ambiente (L-AND-R) fino a raggiungere prestazione perfetta e trasferimento in ambiente (L-OR-R). **(b)** Reti P&CC in ambiente immutabile (linea tratteggiata verde) hanno livelli simili di modularità dei più alti livelli prodotti da MVG (linea in grassetto blu). La combinazione di MVG e P&CC in livelli di modularità ancora più elevati (linea in grassetto verde) mostra che sono più forti che da sole.

Le reti modulari evolvono cosi, perché la loro connettività sparsa ha costi inferiori, ma aiuta anche prestazioni ed evolvibilità perché il problema è modulare. Minimizzare i costi insieme con altre forze comporta aumento di modularità. Si hanno livelli più alti di modularità combinando P&CC e ambienti MVG (Fig. 19b, linea in grassetto vs linea verde tratteggiata).
Nel complesso, P&CC (con o senza MVG) ha livelli di modularità simili tanto quanto più MVG è forte e tanto più quando i rate di cambiamento ambientale sono lenti a causa proprio di MVG forte (Fig. 19b, linee verde vs linea blu continua). La modularità P&CC è superiore a PA anche su problemi non-modulari (Fig. 20a). E' più bassa dei problemi modulari ($p = 0.0011$, retina modulare vs retina non modulare).

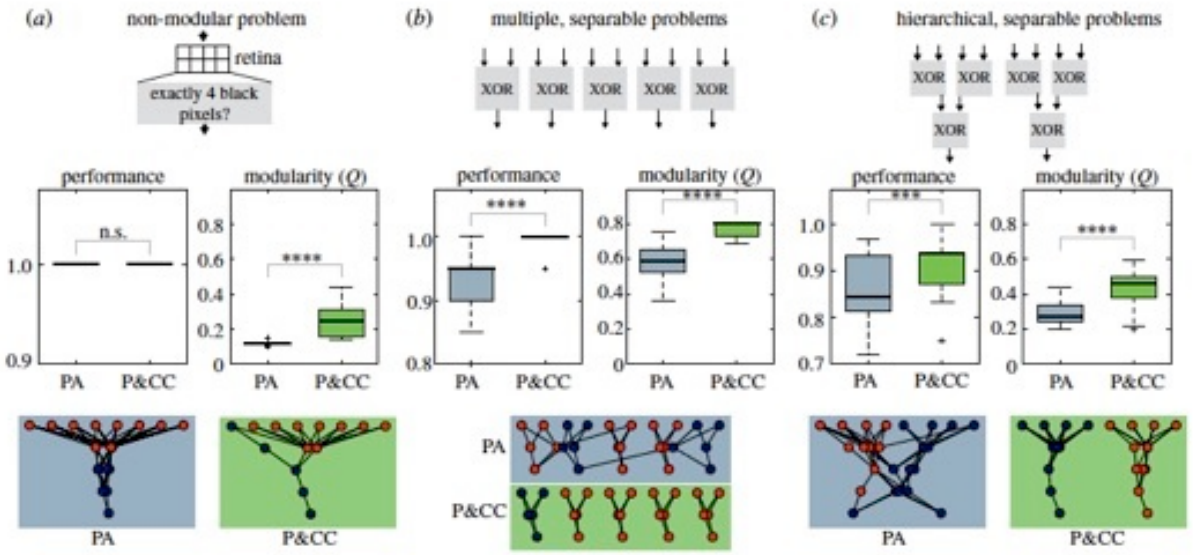


Fig. 20. Risultati con diversi problemi ambientali. **(a)** Problema non-modulare, modularità maggiore con P&CC, inferiore per problemi modulari. **(b, c)** P&CC esegue meglio, più modulare e migliore scomposizione funzionale di PA quando le reti evolvono per risolvere 5 funzioni XOR separate e funzioni XOR gerarchicamente nidificate. Tre/quattro asterischi: valori di *p* inferiori a 0.001 e 0.0001, n.s. nessuna differenza significativa.

Prestazioni e modularità sono più alte anche con una funzione di costo basata su numero di connessioni P&CC-NC (Fig. 21) ma non sono più alte semplicemente perché è usato un secondo obiettivo.

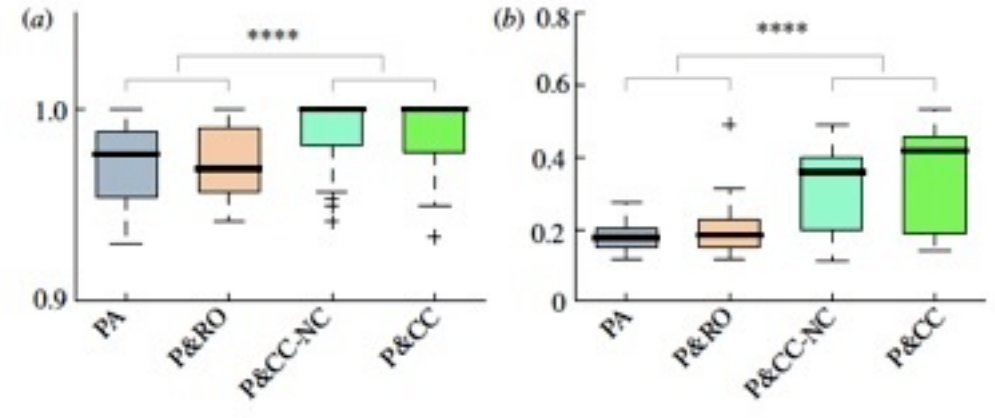


Fig. 21. Funzioni di costo alternativi. **(a)** Prestazione e **(b)** modularità superiori (p< 0.0001) con funzione di costo sulla lunghezza (P&CC) o sul numero di connessioni (P&CC-NC) rispetto PA o prestazione e un obiettivo casuale (P&RO). P&RO ha assegnato un numero casuale invece di un punteggio di costo e massimizzato quel numero casuale.

Tutti questi risultati sono qualitativamente simili in un modello diverso: reti che evolvono per risolvere Task Logici Booleani. Sono stati testati due problemi completamente separabili:
1) con 5 moduli logici XOR (Fig. 20b)
2) con moduli XOR gerarchicamente nidificati (Fig. 20c).
I risultati hanno prodotto una prestazione P&CC anche più alta (Fig. 20b, c) e una correlazione inversa fra costo e modularità, confermando ancora una volta che i costi migliorano i rate di adattamento e le prestazioni alte e, reti a basso costo sono modulari.
Concludendo l'ipotesi della selezione per minimizzare i costi causa modularità anche in ambienti immutabili è confermata, considerando qualsiasi componente della modularità che sorge per minimizzare i costi. Dato che la modularità risultante produce possibilità di evoluzione, minimizzare i costi può servire come processo di avvio automatico che crea modularità iniziale e poi ulteriormente sostenuta da una selezione per evolvibilità. All'inizio sono ipotesi necessarie perché l'evolvibilità non può agire fintanto c'è abbastanza modularità da aumentare la velocità di adattamento.

3.3

Metodi e Modello

Per i modelli di rete si considerano reti feed-forward quindi con nodi disposti in livelli. Il numero massimo di nodi per livello nascosto è: 8/ 4/ 2 per i 3 livelli nascosti nel problema della retina, 8 per singolo livello nascosto nel problema con 5 moduli XOR e 8/ 4/ 4 per i 3 livelli nascosti del problema XOR gerarchico. I possibili valori per i pesi di connessione sono numeri interi -2, -1, 1, 2. I possibili valori per le soglie sono -2, -1, 0, 1, 2. L'informazione passa attraverso la rete in step di tempo discreto un livello per volta. L'uscita di ogni nodo è una funzione dei suoi input:

$$y_j = \tanh\left(\lambda\left(\sum_{i \in I_j} w_{ij} y_i + b\right)\right)$$

yj è l'uscita del nodo *j*, *I* insieme di nodi connessi a *j*, *wij* forza della connessione fra nodo *i* e nodo *j*, *yi* uscita del nodo *i*, *b* è una soglia che determina a quale valore di input le transizioni di output passano da negativo a positivo.
tanh(x) è una funzione di trasferimento che assicura un range di uscita [-1, 1]. λ (qui 20) determina la pendenza della transizione fra questi estremi inibitorio e eccitatorio.
Per quanto riguarda l'algoritmo evolutivo, è basato sulla ricerca di algoritmi ispirati all'evoluzione che contemporaneamente ottimizzano diversi obiettivi (multiobiettivo).
Il modello è rappresentato in Fig. 22:

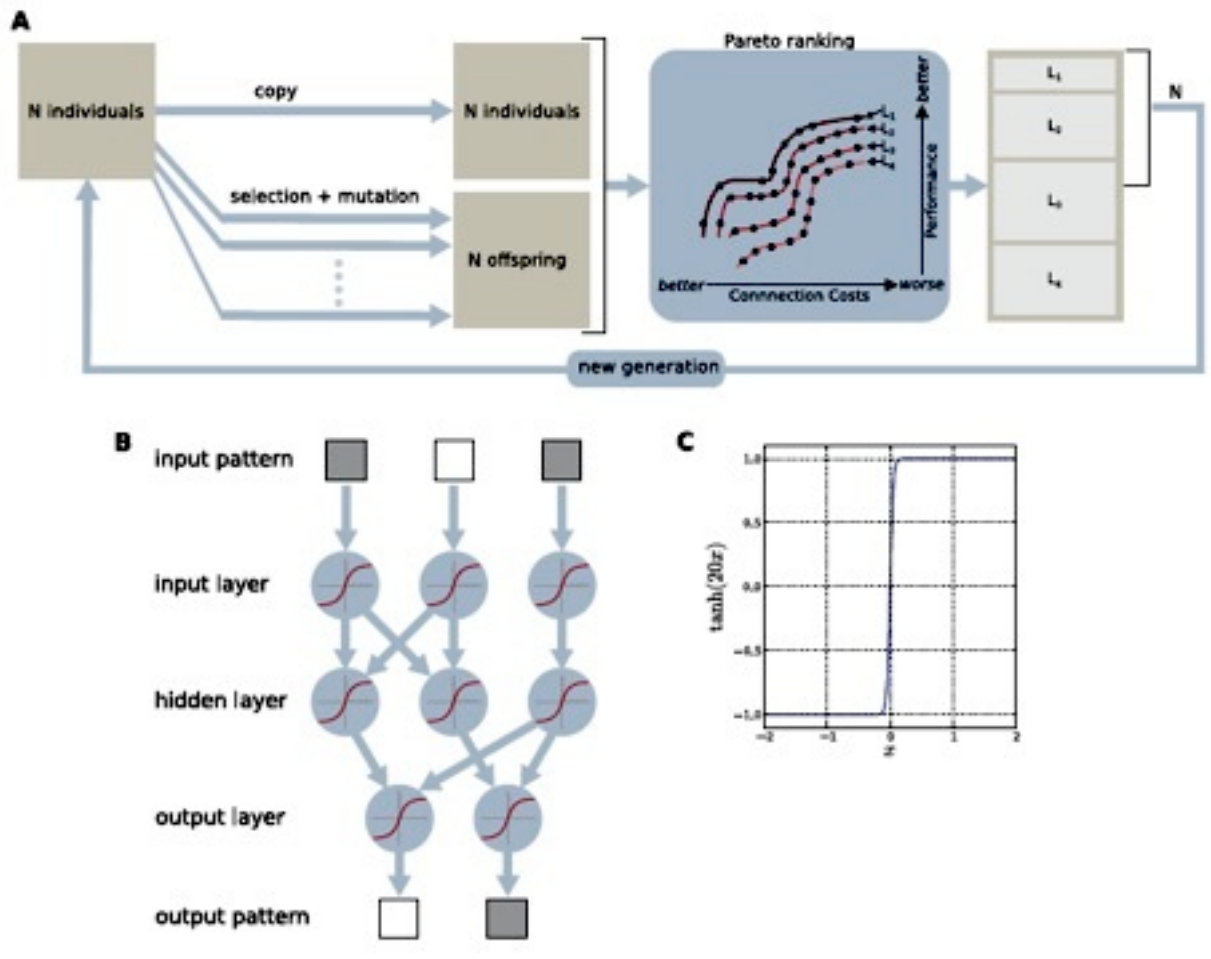


Fig. 22. Modello. **(a)** Algoritmo (NSGA-II). Si inizia con una popolazione di *N* individui generati a caso, viene generata una popolazione prole di *N* individui dai migliori individui della popolazione corrente. L'unione prole e popolazione corrente è sistemata secondo la dominanza stocastica di Pareto (rappresentati da organismi in diversi livelli collegati da linee etichettate L_1, L_2, ecc) e i migliori *N* individui formano la generazione successiva. **(b)** Esempio modello di rete. L'informazione entra nella rete quando è rilevata come un pattern di input. I nodi rappresentano componenti della rete e rispondono attivando o inibendo altri componenti a vari gradi. La forza delle interazioni fra due nodi è rappresentata dal peso della loro connessione, è un valore scalare e, se l'interazione è eccitatoria o inibitoria dipende dal peso positivo o negativo. Tutti i pesi non zero sono rappresentati da una freccia. I segnali entranti in ciascun nodo passano attraverso una funzione di trasferimento per determinare l'output per quel nodo. L'output poi attraversa ognuna delle connessioni uscenti del nodo e dopo esser stato scalato dal peso della connessione uscente serve come componente del segnale entrante per il nodo alla fine della connessione. **(c)** Funzione di trasferimento per ogni nodo. La somma del segnale entrante *x* per un nodo moltiplicata per 20 prima di esser passata alla fdt *tanh(x)* la quale assicura un output nel range [-1, 1]. Moltiplicando per 20 si rende la transizione ripida e simile alla funzione gradino.

L'algoritmo multi obiettivo usato è il *Nondominated Sorting Genetic Algorithm*, versione II (NSGA-II). Si basa sul concetto di dominanza di Pareto: un individuo x^* è detto dominare un altro individuo x se le condizioni 1 e 2 sono vere:
1) x^* non è peggiore di x rispetto a qualsiasi obiettivo;
2) x^* è strettamente migliore di x rispetto ad almeno un obiettivo.
Il primo obiettivo (prestazione) è più importante del secondo (ottimizzazione dei costi di connessione). E' usata una versione stocastica della dominanza in cui il secondo obiettivo è preso in considerazione solo con una data probabilità p. Valori più bassi di p causano più bassa pressione selettiva sul secondo obiettivo. I risultati sono robusti per valori alternativi di p, fino a 1.0 per ambienti statici e 0.95 per ambienti variabili. Questa applicazione stocastica del secondo obiettivo è implementata cosi:
sia r un numero casuale in [0, 1], p probabilità di considerare il secondo obiettivo.
Una soluzione x^* si dice dominare stocasticamente un'altra soluzione x, se una delle due condizioni è vera:
1) $r > p$, x^* è migliore di x rispetto al primo obiettivo;
2) $r <= p$, x^* non è peggiore di x rispetto a entrambi gli obiettivi ma x^* è migliore di x rispetto ad almeno un obiettivo.
La dominanza Stocastica di Pareto è utilizzata due volte nell'algoritmo:

- per selezionare un genitore per la generazione successiva, due individui x_1 e x_2 sono scelti a caso dalla popolazione corrente; se x_1 domina stocasticamente x_2, allora è selezionato x_1; se x_2 domina stocasticamente x_1 allora è selezionato x_2. Se nessuno dei due domina l'altro, l'individuo selezionato è quello che si trova nella parte meno affollata dello spazio obiettivo.
- Per classificare gli individui, prima si identifica l'insieme di soluzioni stocasticamente non dominate e poi si chiamato il primo layer di Pareto; questi individui sono poi rimossi e la stessa operazione ripetuta per identificare livelli successivi.

In ogni generazione, ogni nuova rete è cambiata a caso con quattro possibili tipi di mutazione che non si escludono a vicenda:

- ogni rete ha una probabilità del 20% di avere una singola connessione aggiunta.
- ogni rete ha un 20% di probabilità che una singola connessione casualmente selezionata venga rimossa;
- ciascun nodo della rete ha una probabilità del 4,16% (1/24) di avere la soglia incrementata o decrementata, con stessa probabilità; cinque valori disponibili (-2, -1, 0, 1, 2), mutazioni con valori superiori o inferiori a questi saranno ignorati;
- ciascuna connessione ha probabilità di essere incrementata o decrementata del 2.0/n, con n numero totale di connessioni.

Avere più eventi mutazionali che rimuovono connessioni invece di aggiungerle potrebbe anche produrre reti modulari sparsamente connesse, quindi si ripete con rate di mutazione a vari livelli. Con eventi *remove-connection* con una grandezza più probabile di eventi di tipo *add-connection* non si producono reti modulari.
In ogni esperimento, modularità e prestazione P&CC con valori di mutazione di default sono stati maggiori di PA con valori di rate di mutazione polarizzata. I risultati sono qualitativamente gli stessi quando pesi e polarizzazioni sono numeri reali anziché interi e mutati attraverso una

perturbazione gaussiana. I nodi non sono mai aggiunti né rimossi e quelli senza connessioni non sono visualizzati o inclusi nei risultati.
L'algoritmo MOLE serve per trovare soluzioni con alte prestazioni e diverse combinazioni di modularità e lunghezze. E' un'ottimizzazione di ricerca multi obiettivo con due obiettivi:

- il primo obiettivo dà priorità agli individui con alta prestazione;
- il secondo dà priorità a individui lontani da altri individui già scoperti dall'algoritmo e, tale distanza misurata su un asse cartesiano con costi sull'asse *x* e modularità sull'asse *y*.

Questi algoritmi hanno dimostrato di esplorare meglio lo spazio di ricerca perché meno suscettibili a rimanere bloccati in ottimi locali. A differenza di un algoritmo evolutivo tradizionale progettato solo per un tipo di soluzione con un gradiente di fitness verso quel tipo di soluzione, MOLE ricerca soluzioni con alte prestazioni per ogni possibile combinazione modularità/costo. Non garantisce di trovare la soluzione ottimale per ogni punto ma un campionamento statistico delle probabilità di poter scoprire una soluzione di alta qualità in ogni area dello spazio di ricerca. Se tale spazio è piccolo, i valori possono essere determinati controllando ogni possibile soluzione (approccio non valido per questi problemi).
Per il problema della retina, il numero di pesi possibili è: (8 x 8) + (8 x 4) + (4 x 2) + (2 x 1) +23 = 129 dovuto al numero massimo possibile di nodi dei livelli di input, nascosto e output e la polarizzazione per ciascuno dei 23 possibili nodi. Ognuno dei pesi può essere uno dei quattro valori o uno zero se non esiste collegamento e, le polarizzazioni uno dei cinque valori con un numero di soluzioni pari a $5^{129} = 10^{90}$.
Inoltre risulta impossibile cercare esaustivamente lo spazio per cui è stato campionato casualmente per vedere se è possibile trovare soluzioni a più alta prestazione con una varietà di costi e livelli di modularità ma non si arriva a soluzioni con alta prestazione. Per ottenere 50 prove, ognuna con una rete con prestazione perfetta in ambienti L-AND-R/L-OR-R, sono prese 110 e 116 prove per P&CC, 320 e 364 prove per PA. 1000 cloni di ciascuna delle reti evolute nell'ambiente si alternano, fino ad arrivare a una prestazione perfetta oppure dopo 5.000 generazioni. Esistono nodi a due dimensioni (*x, y*). Le coordinate geometriche input e output per tutti i problemi sono fissate durante l'evoluzione. Per tutti i problemi, gli input hanno valori di *y* pari a 0. Per il problema della retina, i valori *x* per gli input: - 3.5, -2.5, …, 3.5 e l'output: 4, 0. Per il problema con i 5 moduli XOR, i valori *x* per gli input: - 4.5, - 3.5, …, 4.5 e gli output hanno tutti valori *y* di 2 con valori *x* di (- 4, - 2, 0, 2, 4). Per il problema con funzioni XOR scomponibili gerarchicamente nidificate, i valori *x* per gli input: - 3.5, - 2.5, …, 3.5 e gli output hanno tutti valori *y* di 4 con valori *x* di - 2, 2.
La posizione geometrica dei nodi è consequenziale solo quando vi è un costo superiore (P&CC).

CAPITOLO 4

Evoluzione spontanea della modularità [*]

Come già evidenziato, sistemi biologici e artificiali hanno caratteristiche generali di progettazione, mostrano modularità, come la separabilità di progettare in unità che eseguono in maniera indipendente, almeno in prima approssimazione. Inoltre, mostrano riuso di alcuni pattern di circuiti, chiamati motivi di rete in diverse parti del sistema. Queste caratteristiche permettono la costruzione di sistemi complessi utilizzando semplici blocchi.
Una caratteristica nota dei modelli computazionali di evoluzione biologica è che i sistemi evoluti di solito sono intrinsecamente cablati e non modulari. Le soluzioni non modulari sono spesso meglio ottimizzate, infatti la ragione di tale mancanza di modularità in reti evolute è che le strutture modulari sono meno ottimali di quelle non modulari. Ci sono molte possibili connessioni che rompono la modularità e aumentano la fitness.
Anche una soluzione inizialmente modulare evolve rapidamente in una delle tante possibili soluzioni non modulari, per cui la mancanza di modularità è uno dei motivi dell'evoluzione computazionale. Attualmente può generare design solo per task semplici, è difficile per complessità elevate.
Diversi studi (Calabretta e al., 1998) hanno suggerito che la duplicazione di sottosistemi o una selezione per stabilità/ robustezza possa promuovere la modularità.
Lipson e al. (2002) hanno suggerito che la modularità possa sorgere spontaneamente in ambienti mutevoli, ossia design con più modularità hanno maggiore capacità di adattamento e quindi maggiori rate di sopravvivenza in ambienti che cambiano. Però l'evoluzione computazionale in ambienti che cambiano casualmente non sono sufficienti a produrre modularità.
La caratteristica fondamentale evidenziata da questo studio [KAL05] è l'evoluzione in un ambiente (goal evolutivo) che cambia nel tempo in maniera modulare, ripetendolo con diversi goal, ciascuno con una diversa combinazione di sottogoal. Tale *modularly varying goals* MVG porta all'evoluzione spontanea di una struttura modulare.

[*] N. Kashtan, U. Alon "Spontaneous evolution of modulariy and network motifs" in PNAS, Vol.102, N.39, 2005.

4.1

Metodi: evoluzione di un circuito elettronico e una rete neurale

Come modello, consideriamo circuiti elettronici rappresentati da un genoma binario con un numero fisso di geni che codificano gate NAND, un gene per ogni output. Ogni generazione dei migliori L circuiti passano invariati alla generazione successiva. Ogni circuito è poi mutato in maniera casuale (probabilità di mutazione $P_m = 0.7$ per genoma). Una penalità di fitness di 0.2 per ogni gate su un numero predefinito di gate efficaci (11 gate per i circuiti in Fig. 23). Per "gate efficaci" si intende porte con percorso diretto all'output. La dimensione della popolazione, $S = 1000$, $L = 300$ (Fig. 26), $S = 2000$, $L = 500$ (Fig. 23).

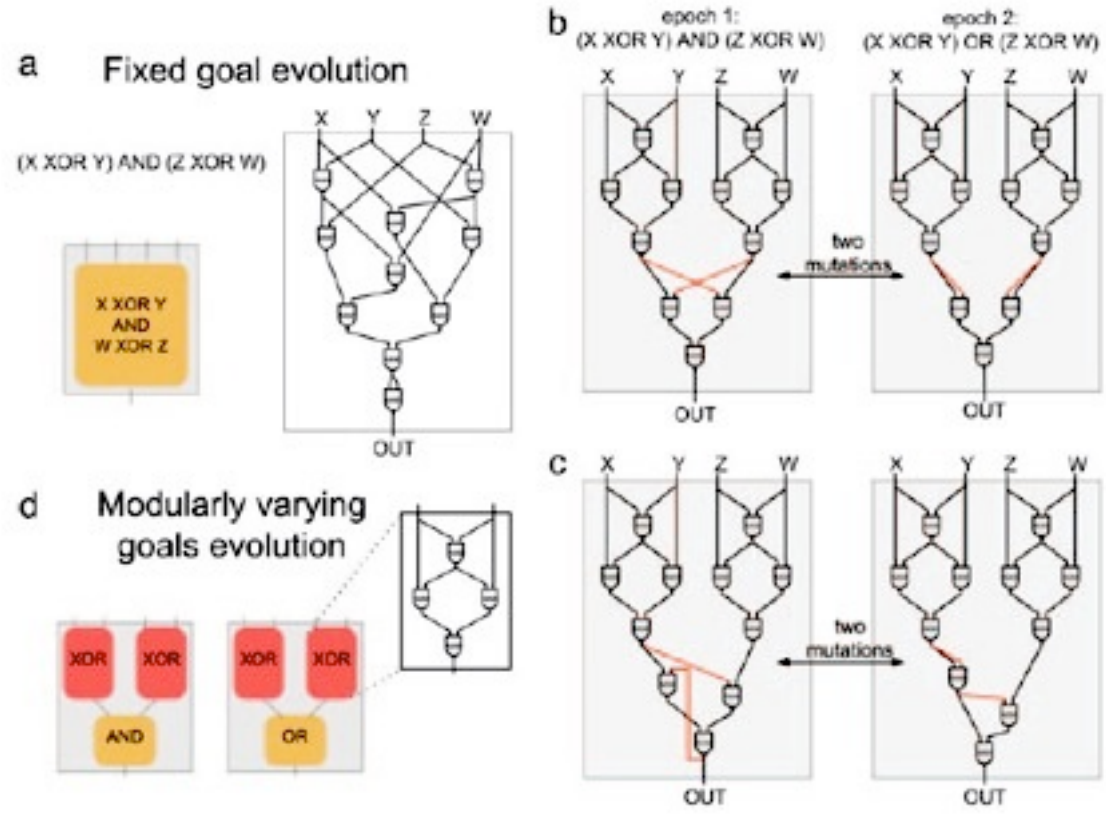


Fig. 23. Evoluzione circuiti con goal fisso e *modularly varying goal*. **(a)** Evoluzione circuito con goal fisso verso G1. Soluzioni non modulari simili trovate per G2. **(b)** Evoluzione circuiti con *modularly varying goal*. Rosso: connessioni ricablate quando il goal cambia. **(c)** Evoluzione circuito con *modularly varying goal*, mostra feedback fra due dei 3 gate NAND più bassi. **(d)** Struttura modulare dell'evoluzione dei circuiti *modularly varying goal* con 2 moduli XOR con input in un terzo modulo che implementa una funzione AND/OR, a seconda del goal. Ogni modulo XOR ha 6 nodi (2 di input e 4 gate interni) e mostra 2 cicli feed-forward.

Risultati simili sono stati trovati sotto un range di parametri, cosi come popolazioni più grandi e rate di mutazione più piccoli e con operatori di crossover e mutazione. Il genoma della rete di dimensione fissa di 15 geni ciascuno che codifica un neurone. I neuroni in quattro layer con 8, 4, 2 e un neurone per layer. Le connessioni solo fra layer vicini in maniera feed-forward. Le connessioni della retina solo al primo layer. L'output è il neurone al quarto layer. A ciascun neurone è assegnato un numero massimo di connessioni entranti: 3 input per un neurone nel primo, secondo e terzo layer, 2 input per un neurone nel quarto layer. Ogni connessione con pesi -1 o 1. Se due degli input sono dallo stesso neurone, allora il peso reale della connessione è la somma dei pesi delle due

connessioni. Una penalità di 0.01 è applicata a ogni neurone supplementare su un numero predefinito di neuroni (qui 13). Le mutazioni e crossover sono usate come operatori di evoluzione con $S = 600$ e $L = 150$. La probabilità di crossover e di mutazione *(*0.5) per genoma.

4.2

Casi di studio

Come casi di studio, consideriamo due modelli di sistema: il primo costituito di circuiti logici combinatori costituiti di un singolo tipo di gate, NAND (funzione NOT-AND) con più input, etichettati X, Y, Z, W. I gate NAND sono universali, nel senso che ogni funzione logica può essere implementata dal circuito composto di porte NAND. Il goal dell'evoluzione è una funzione logica *G*. La fitness è la frazione di volte che il circuito fornisce l'output corretto *G*, valutata su tutte le possibili combinazioni di valori booleani di input. E'applicato un algoritmo standard evolutivo per cercare un circuito che massimizza la fitness e poi raggiunge il goal. Si inizia con una popolazione di genomi casuali che rappresentano circuiti cablati a caso. Per ogni generazione, ogni circuito ha una probabilità *preset* (rate di mutazione) di cambiare a caso una delle connessioni fra i suoi gate.
I circuiti con fitness bassa vengono rimossi da parte della popolazione, quelli con fitness alta sono replicati con dimensione fissa di popolazione. Il processo è poi ripetuto, i circuiti prima evoluti verso goal fisso, con XOR funzione or-esclusivo:

G1 = (X XOR Y) AND (Z XOR W)

La fitness aumenta nel tempo giungendo a una soluzione perfetta in 10^5 generazioni in 36 dei 50 esperimenti. Una soluzione perfetta è ottenuta in 9.000 generazioni (Fig. 24a, fitness vs generazioni).

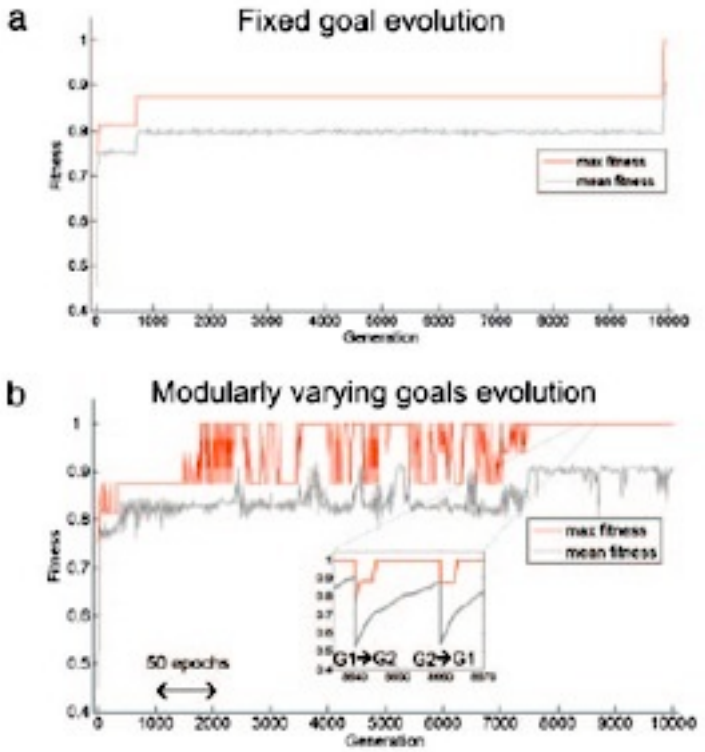


Fig. 24. Evoluzione di circuiti con *modularly varying goal* e goal fissi. **(a)** Fitness come funzione di generazioni con goal fisso G1. Rosso: miglior circuito nella popolazione; grigio: fitness media. **(b)** Fitness come funzione di generazioni con evoluzione *modularly varying goal* verso G1 e G2. Il goal attivato ogni 20 generazioni. La fitness mostrata nella risoluzione di 20 generazioni, appena prima dell'evento di cambio di ogni goal. (All'interno) Zoom della fitness attorno a due eventi che cambiano, la fitness scompare e ricompare dopo ogni cambio di goal.

Circuiti diversi raggiungono soluzione perfetta con 10 gate. Soluzione tipica mostrata in Fig. 23a. Nonostante il fatto che il goal G1 possa essere scomposto in sottoproblemi (2 XOR e 1 AND), l'architettura dei circuiti evoluti è sempre non modulare. Reti non modulari hanno $Q_m \approx 0$, reti modulari hanno valori di Q_m fra 0.3 e 1. I circuiti evoluti mostrano modularità molto più bassa: $Q_m = 0.12 + - 0.02$.
Poi è stata eseguita l'evoluzione in cui il goal cambia periodicamente fra due funzioni diverse che abbiano però sottoproblemi condivisi (approccio *modularly varying goals*"). Si usa il goal G1 per un epoca di 20 generazioni poi commutato in una funzione simile G2 in cui due computazioni XOR sono collegate da un OR invece che da un AND:

G2 = (X XOR Y) OR (Z XOR W)

I goal cambiati continuamente ogni 20 generazioni arrivando a una soluzione perfetta in tutti i 50 esperimenti in <10.000 generazioni. Quindi i circuiti evoluti sono in grado di adattarsi a soluzioni perfette per ciascuno dei due goal. Ogni volta che il goal cambia, la popolazione raggiunge una soluzione perfetta per il nuovo goal in cinque generazioni e rimane fino al cambio successivo. In generale, soluzioni in grado di evolversi durano per molte epoche (Fig. 24b).
La struttura dei circuiti evolvibili è altamente modulare e la modularità è evidente e quantificabile da una misura significativa di modularità $Q_m = 0.54 + - 0.02$.
In (Fig. 26 b, c) sono mostrati due diversi esempi. Le soluzioni sono costituite di due moduli identici, ciascuno esegue una computazione XOR e un terzo modulo che esegue AND o OR sull'output degli XOR. I moduli XOR, ciascuno fatto di 4 gate NAND. La differenza fra le soluzioni perfette per i due goal differisce per due connessioni e questo spiega come la popolazione può rapidamente adattarsi quando il goal cambia (Fig. 24b).
Le soluzioni evolvibili sono più grandi delle soluzioni con goal fisso e di solito usano 11 gate, al contrario di 10 in soluzioni evolute con goal fisso. Circuiti che evolvono con goal che variano sono non modulari. A questo scopo, sono usate funzioni logiche casuali come goal e commutate fra loro come descritto, trovando che reti evolute in generale hanno architettura non modulare. Queste reti sembrano "dimenticarsi" del vecchio goal e cercano di trovare una soluzione al nuovo goal da zero. Lo stesso vale per i casi in cui un problema è G1 o G2 e l'altro problema è una funzione logica casuale a quattro input.
Successivamente, sono stati eseguiti esperimenti dove l'evoluzione inizia con un circuito modulare ma con goal fisso. La modularità diminuisce rapidamente in poche generazioni e anche con una leggera selezione per la dimensione piccola del circuito (Fig. 25). Questo risultato evidenzia il ruolo di *modularly varying goals* nel preservare la modularità a fronte di circuiti non modulari più ottimali.

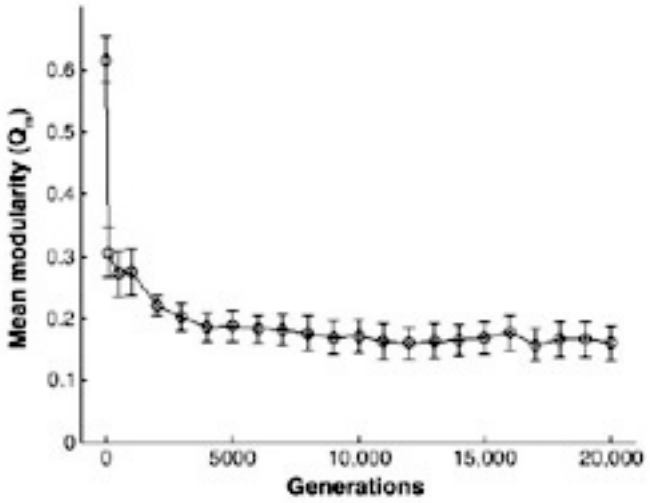


Fig. 25. Circuito modulare all'inizio perde rapidamente modularità con goal fisso. Ogni esperimento inizia con popolazione di circuiti modulari identici con fitness perfetta per il goal G2. A generazione zero, la popolazione è posta sotto evoluzione con goal fisso con stesso goal G2 con una pressione di selezione per le piccole dimensioni del circuito (una penalità di fitness di 0.2 per ogni gate addizionale sopra il decimo gate). La misura della modularità media (+ - SE) vs generazioni di circuiti con fitness migliore. Le statistiche sono per 20 esperimenti indipendenti, con 4 diversi circuiti modulari iniziali che soddisfa G2.

Stesse conclusioni per circuiti più grandi e problemi più complessi. Per esempio, l'evoluzione di un problema con 6 input, 3 output con *modularly varying goals* produce circuiti con architettura modulare (Fig. 26).

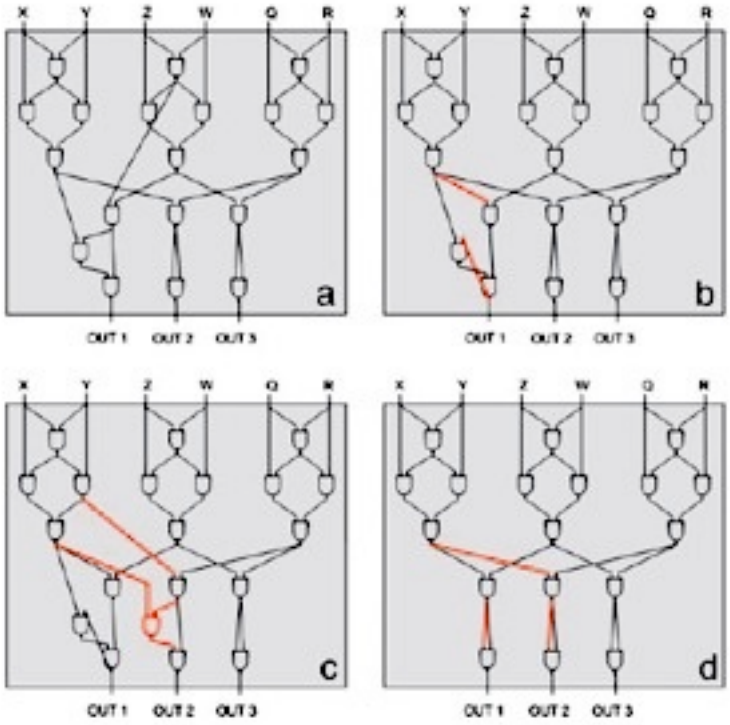


Fig. 26. Circuito elettronico con 6 input, 3 output evoluti con *modularly varying goal* con 4 goal: G1 = (A OR B; A AND C; B AND C); G2 = (A AND B; A OR C, B AND C); G3 = (A AND B; A AND C; B OR C); e G4 = (A AND B; A AND C; B AND C), dove A = X XOR Y, B = Z XOR W, C = Q XOR R.
Il goal cambia ogni 20 generazioni nell'ordine G1, G4, G2, G4, G3, G4, G1, G4, ecc. Soluzioni perfette che cambiano rapidamente fra i goal evoluti sono trovate sotto i goal G1, G4, G2 e G4, rispettivamente. Linee e gate in rosso rappresentano cambiamenti in fase di adattamento a ogni nuovo goal.

Questi circuiti sono stati modificati includendo 3 moduli separati che calcolano gli XOR, cosi sono stati capaci di evolvere per risolvere ciascuno dei 4 MVG.
Tale approccio è applicato anche a un secondo modello computazionale già presentato nel capitolo 3 però qui esposto a condizioni diverse con goal fisso ed MVG. Il modello è quello del riconoscimento di pattern usando reti neurali, con un mapping genotipo-fenotipo che assicurasse che le reti fossero di tipo feed-forward con 4 layer di nodi, collegate da archi con peso. Ogni nodo somma i suoi input pesati e attiva i suoi output se la somma supera una soglia.

4.3

Pattern Recognition

Nel problema di *pattern recognition*, i neuroni ricevono input da una retina di 4 pixel di larghezza per 2 di altezza. Ciascun pixel può assumere valore 0 o 1. Il goal è riconoscere oggetti sulla parte sinistra e destra della retina (Fig. 27a). Un oggetto sinistro o destro è presente se i 4 pixel a sinistra o destra corrispondono a uno dei pattern di un insieme predefinito.

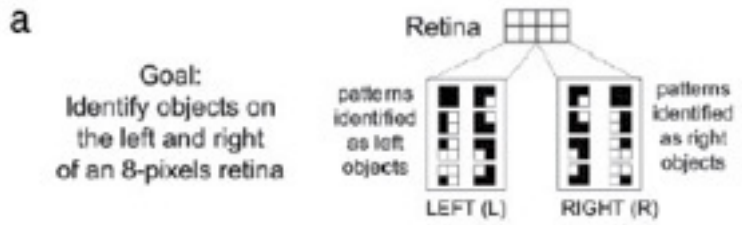


Fig. 27. Evoluzione di reti per un task di pattern recognition con goal fissi e MVG. **(a)** Goal: riconoscere oggetti nelle parti sinistra e destra di una retina 4 x 2. Un oggetto sinistro (o destro) esiste se il quarto pixel sinistro (o destro) corrisponde a uno dei pattern di un insieme predefinito. Un oggetto sinistro è definito da 3 o più pixel neri oppure 1 o 2 pixel neri nella sola colonna di sinistra. Un oggetto destro è definito in modo simile, con 1 o 2 pixel neri nella sola colonna di destra. Goal: identificare la presenza di oggetti su entrambi le parti della retina (L AND R).

La rete deve decidere se l'input si inserisce in una certa combinazione di oggetti a sinistra e destra (per es. su entrambi i lati). L'output è un nodo con valore 0 o 1. Per evolvere le reti è usato un algoritmo genetico standard con crossover e mutazioni. L'ambiente contiene 100 diversi pattern della retina scelti a caso. Iniziando con una popolazione di genomi casuali, in ogni generazione è stato eseguito un crossover fra due genomi di rete scelti casualmente con probabilità P_c. Inoltre, una connessione, peso o soglia in ogni rete è stato modificato con probabilità *preset* mutazionale P_m. La fitness della rete è la frazione di riconoscimenti corretti nell'ambiente. Reti con fitness maggiore sono state replicate. Nell'evoluzione con goal fisso, soluzioni quasi perfette (almeno 95% di riconoscimento corretto) sono raggiunte dopo 21.000 generazioni. Al contrario, la struttura delle reti evolute era non modulare (Fig. 27b) ($Q_m = 0.15 + - 0.02$).
Come nel caso dei circuiti elettronici, l'evoluzione con goal fisso produce una rete non modulare anche se il problema stesso è facilmente scomponibile in sottogoal separati. E' stata eseguita l'evoluzione modulare con MVG scambiando due goal con diverse combinazioni di sottogoal definiti su ciascuna metà della retina (Fig. 27c). I goal cambiano ogni 20 generazioni e l'evoluzione fornisce rapidamente reti quasi perfette con architettura modulare ($Q_m = 0.35 + - 0.02$) in 2.800 generazioni.

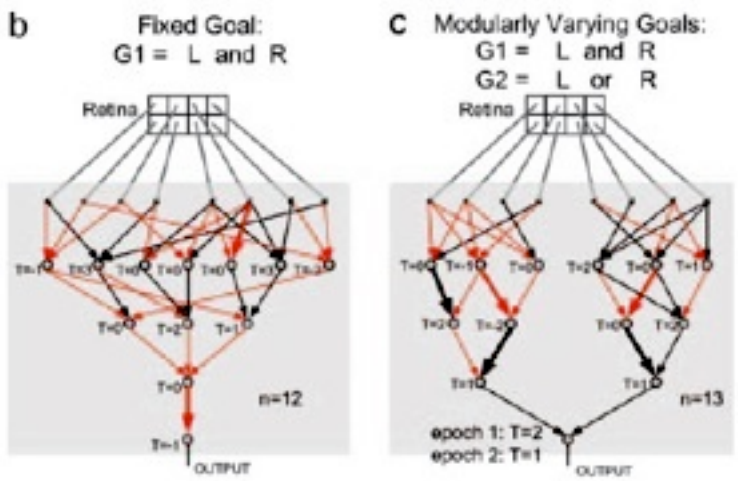


Fig. 27. (b) Evoluzione rete con goal fisso. Linee nere/rosse: pesi positivi/negativi. Linee in grassetto: pesi doppi. **(c)** Evoluzione rete con MVG con 2 goal: L AND R, L OR R. La rete può adattarsi rapidamente ogni volta che il goal cambia, modificando un'unica soglia del neurone più basso (da $t = 2$ al primo goal a $t = 1$ al secondo goal). In b e c, n indica il numero di neuroni (nodi) della rete.

Due moduli distinti (non identici) evoluti spontaneamente nella rete, ognuno che controlla una parte diversa (Fig. 27 e).

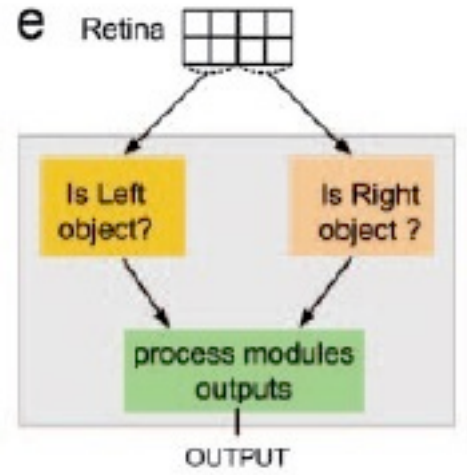


Fig. 27. (e) Struttura modulare della rete evoluta con MVG. Due moduli distinti, ciascuno controlla una parte della retina e un terzo modulo elabora gli output.

Pertanto a differenza dell'evoluzione con goal fisso, la struttura modulare dei problemi è "appresa" e implementata nella soluzione. Reti evolute con MVG sono state in grado di adattarsi a soluzioni quasi perfette per ogni nuovo goal in circa 3 generazioni dopo che il goal è cambiato. Questa evolvibilità è causata dal fatto che reti evolute per i diversi goal differiscono solo leggermente, in molti casi differiscono nel valore di soglia di un singolo neurone, consentendo il passaggio fra reti con singola mutazione (Fig. 27c).

4.4

Risultati

MVG possono produrre evoluzione spontanea di architetture di rete modulari con evidenti motivi di rete. Lo stesso processo evolutivo con goal fisso fornisce soluzioni non modulari e con meno motivi di rete.
Reti evolute con MVG sembrano scoprire sottoproblemi di base comuni ai diversi goal e sviluppano un modulo strutturale distinto per implementare ciascun sottoproblema.
L'evoluzione con MVG produce reti che possono adattarsi rapidamente a ciascuno dei diversi goal solo grazie a pochi cambiamenti. Non tutti gli insiemi di goal che cambiano porta sempre a strutture modulari. L'evoluzione di reti con MVG con nessun sottogoal comune, non sembrano evolvere la struttura modulare e in tali casi quando il goal cambia, le reti necessitano di molto tempo per adattarsi al nuovo e come se l'evoluzione iniziasse da zero. Con MVG l'adattamento al nuovo goal è accelerato dalla presenza di moduli esistenti utili per i goal precedenti.
Come velocizzare l'evoluzione MVG (in termini di numero di generazioni che raggiungono soluzione perfetta) rispetto all'evoluzione con goal fisso?
Una ragione è che l'evoluzione con goal fisso è spesso lenta e la popolazione rimane bloccata in un massimo locale di fitness. Poiché il picco di fitness cambia ogni volta che il goal cambia, MVG può aiutare la popolazione a uscire da queste trappole locali.
Nel caso di molti goal che cambiano, MVG sembra guidare la popolazione verso una regione di spazio contenente picchi di fitness vicini per ognuno dei goal e questa regione sembra corrispondere a reti modulari. Oltre alla struttura modulare, le reti evolute con MVG mostrano importanti motivi di rete che vengono riutilizzati in ogni rete in diversi moduli.
Una possibile spiegazione dell'origine dei motivi in reti evolute è che le reti modulari sono localmente più dense di reti non modulari della stessa dimensione e connettività. Questa densità locale tende ad aumentare il numero di sottografi.

CAPITOLO 5

Modularità da specializzazione [*]

La modularità può migliorare la capacità di generare variazioni ereditarie delle capacità di adattamento, per due motivi:

- l'organizzazione in moduli consente cambiamenti all'interno di un modulo senza perturbare gli altri;
- i moduli possono essere combinati e riutilizzati per creare nuove funzioni.

Abbiamo considerato due scenari per l'origine e il mantenimento di moduli:

- uno è la combinazione di selezione direzionale e selezione stabilizzante;
- l'altro coinvolge MVG.

Un ambiente che oscilla in modularità costituisce l'alternativa ad ambienti che evolvono con obiettivi simili, per cui tali fluttuazioni modulari possono essere sufficienti a produrre e mantenere modularità.

La modularità può sorgere in reti di regolazione genica come sottoprodotto della specializzazione dell'attività dei geni. Tale specializzazione include l'evoluzione di pattern di attività nuovi che sorgono in una parte specifica o sotto specifiche condizioni ambientali.

Le reti con un pattern di attività I aumentano la loro modularità quando la selezione favorisce un pattern II con le seguenti condizioni:

- la selezione favorisce il pattern I in modo che le reti evolute producano entrambi i pattern I e II;
- i pattern I e II condividono lo stato di attività di alcuni geni con dei ruoli specifici.

5.1

Modello e Fitness

Il modello è costituito da una rete di N geni. Di ogni gene, lo stato di attività è regolato da altri geni della rete. Il genotipo di un individuo è l'insieme di interazioni fra i suoi geni ed è rappresentato da una matrice A = (a_{ij}) in cui gli elementi diversi da zero indicano attivazione (a_{ij} = 1) o repressione (a_{ij} = - 1) del gene i esercitata dal gene j. Lo stato della rete al tempo t è dato da un vettore $s_t = (s_t^{\,0},\ldots, s_t^{\,N-1})$. Un certo gene i al tempo t può essere attivo ($s_t^{\,i} = 1$) o inattivo ($s_t^{\,i} = -1$). Il cambiamento dell'attività dei geni è modellato dalla seguente equazione, dove $\sigma(x)$ è pari a 1 per $x > 0$, - 1 in tutti gli altri casi.

$$s_{t+\tau}^{i} = \sigma\left[\sum_{j=1}^{N} a_{ij} s_t^{j}\right]$$

[*] C. Espinosa-Soto, A. Wagner "Specialization Can Drive the Evolution of Modularity" in PLoS Comput Biol, N.6, 2010.
R. Calabretta, S. Nolfi, D. Parisi, A. Wagner "Duplication of modules facilitates the evolution of functional specialization" in Artificial Life, N.6, 2000.

Un tratto di fenotipo è un attrattore, un modello di attività del gene stabile risultante da dinamiche di una rete di regolazione genica. Le simulazioni evolutive sono costituite da mutazioni iterative e selezione. Confrontando un insieme di pattern di attività del gene di riferimento per attrattori di rete reali, si ha che reti con attrattori simili ai pattern di attività selezionate producono fitness più alta. Per quantificare la modularità, è usato un algoritmo che identifica i moduli come gruppi di nodi sovrapposti con poche connessioni fra loro. Se i geni di moduli singoli interagiscono con molti geni al di fuori del loro modulo, l'autonomia dei moduli diminuisce.
La funzione di fitness utilizzata confronta un insieme di pattern di attività di riferimento del gene con effettivi attrattori di rete. La misura di fitness include anche la probabilità che un attrattore è raggiunto a dispetto di perturbazioni. In tal modo, non solo si prende in considerazione l'identità di un attrattore, ma anche la sua robustezza, una caratteristica importante per la stabilità.
Per ogni pattern di attività *X* che contribuisce alla fitness e per ogni rete, la valutazione della fitness è costituita dalle seguenti fasi: i) lo stato iniziale della rete al tempo 0 è scelto per essere una perturbazione del pattern *X*, passando da una probabilità in cui lo stato iniziale di ogni gene differisce da *X* di 0,15; ii) vengono eseguite dinamiche di rete (vedi equazione) fino al raggiungimento di qualche nuovo attrattore *Y*; iii) si calcola la distanza di Hamming che separa *Y* da *X*, e si calcola il contributo della fitness di questa traiettoria di sviluppo; iv) si ripetono i passi i) -iii) 500 volte per determinare 500 valori della traiettoria di sviluppo.
Molti di questi valori dovrebbero corrispondere alla stessa condizione iniziale e la distribuzione delle possibili condizioni iniziali è sbilanciata verso pattern di attività simili al pattern di riferimento *X*. Questo riflette l' ipotesi che la selezione favorisce condizioni iniziali simili che portano allo stesso pattern di attività selezionato. Abbiamo anche ipotizzato che pattern di attività simili al pattern di riferimento sono più probabili essere richiesti come condizioni iniziale.
La fitness della rete è calcolata come:

$$f(g) = 1 - e^{-3g}$$

g è la media aritmetica di tutti i valori della traiettoria. Quindi le reti ottimali saranno quelle con dinamiche che portano a diversi attrattori corrispondenti a pattern di riferimento, e non quelli con un unico attrattore che è una combinazione dei pattern di riferimento.

5.1.1

La specializzazione aumenta la modularità

Per vedere se la specializzazione aumenta la modularità sono esaminate 200 popolazioni indipendenti di reti di regolazione genica che evolvono. Ognuna con reti identiche, oggetto di 500 cicli di generazione di mutazioni e selezione verso il raggiungimento di un attrattore punto fisso I.
Il numero di generazioni è scelto per garantire che le reti che raggiungono I stabilmente possano sorgere nella popolazione. Dopo lo sviluppo del pattern I, alla popolazione è permesso di evolvere per più di 1500 generazioni con un pattern II. Le reti più forti emerse sono state quelle capaci di raggiungere stabilmente I e II da diverse condizioni iniziali. Quindi la selezione ha mantenuto la capacità di raggiungere il pattern I e al tempo stesso favorire l'acquisizione del II.
Il pattern II è stato scelto in modo tale che metà dei geni della rete avesse un espressione di stati identica (condivisa) in entrambi e l'altra metà differisse nel proprio stato (Fig. 28A).
Tali pattern di attività scelti ipotizzando che le interazioni fra geni con stati condivisi e il resto dei geni potessero impedire il costante stato di attività della prima o la capacità di quest'ultima di acquisire diversi stati in maniera indipendente da geni con stati costanti. Le interazioni fra diversi insiemi di geni possono essere contro selezionati, risultando così in due insiemi con solo connessioni sparse fra loro. La modularità aumenta successivamente evolvendo verso il raggiungimento di entrambi i pattern, è risulta evidente sia in reti con fitness più alta (Fig. 28B) che su tutte le reti di una popolazione (Fig. 28C).

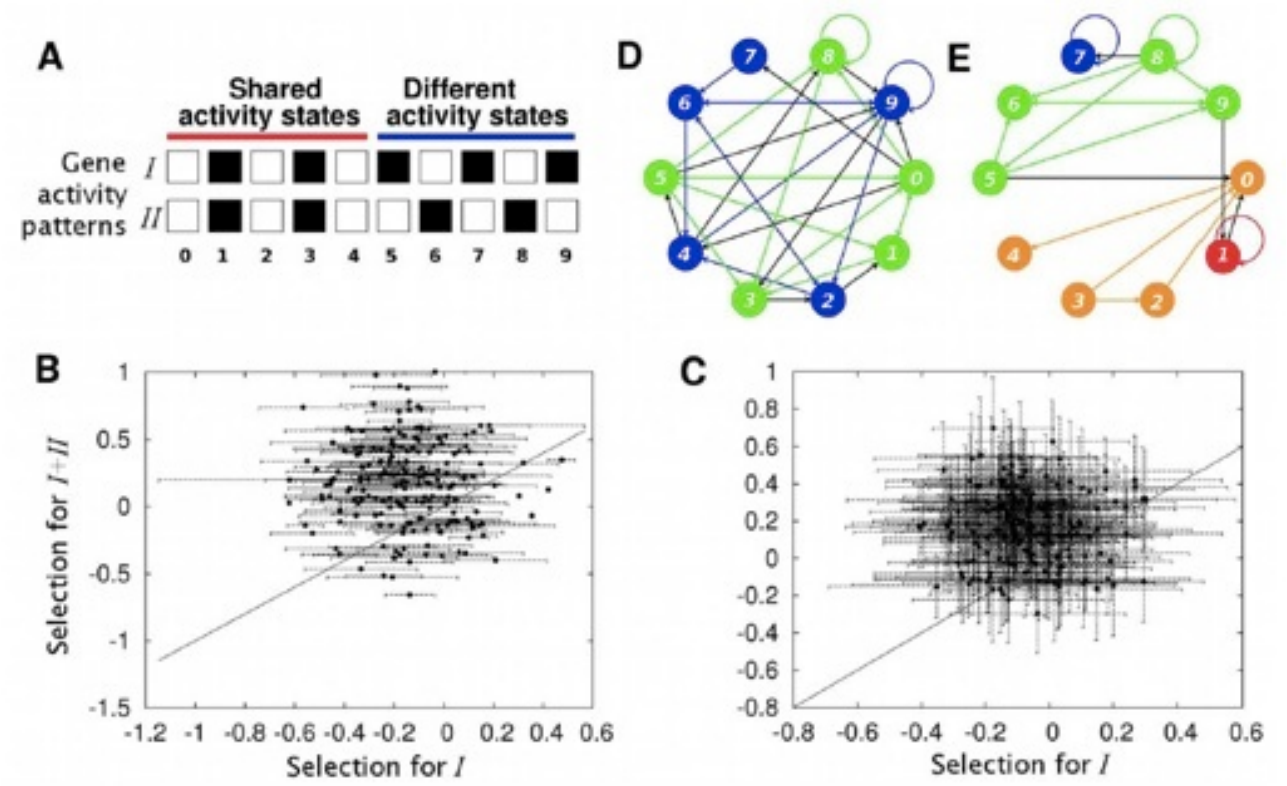


Fig. 28. Aumento di modularità dopo la selezione di un nuovo pattern. **(A)** Pattern I e II condividono lo stato di attività dei geni 0-4 e si differenziano in quello dei geni 5-9. Quadrati bianchi, neri: geni inattivi, attivi. **(B, C)** Asse orizzontale: modularità dopo la selezione del pattern I, asse verticale: modularità dopo la selezione dei pattern I e II. **(B)** Modularità della rete con fitness più alta e **(C)** metà popolazione modulare. I punti al di sopra della diagonale mostrano popolazioni in cui la modularità aumenta dopo la selezione del pattern II. La lunghezza delle linee indica una deviazione standard. **(D, E)** Nodi con stesso colore: geni di uno stesso modulo. Archi neri: interazioni fra geni di moduli diversi. **(D)** Rete con fitness più alta in una popolazione dopo la selezione di I. L'algoritmo di Newman divide la rete in insiemi in cui i geni 0-4 e 5-9 sono mescolati. Questa rete ha una modularità non normalizzata (0.18) e una modularità normalizzata (-0.1). **(E)** Rete con fitness più alta in una popolazione dopo la selezione per pattern I e II. Rete divisa in moduli in cui geni con stati condivisi (geni 0-4) e differenti (geni 5-9) in pattern I e II si trovano a parte. Questa rete ha modularità non normalizzata (0.39) e normalizzata (0.7).

La modularità non aumenta quando la selezione del pattern II è assente, né quando le reti evolvono in assenza di selezione. L'aumento di modularità non è transitorio ma si mantiene circa sullo stesso livello per almeno 10.000 generazioni in più (Fig. 29).

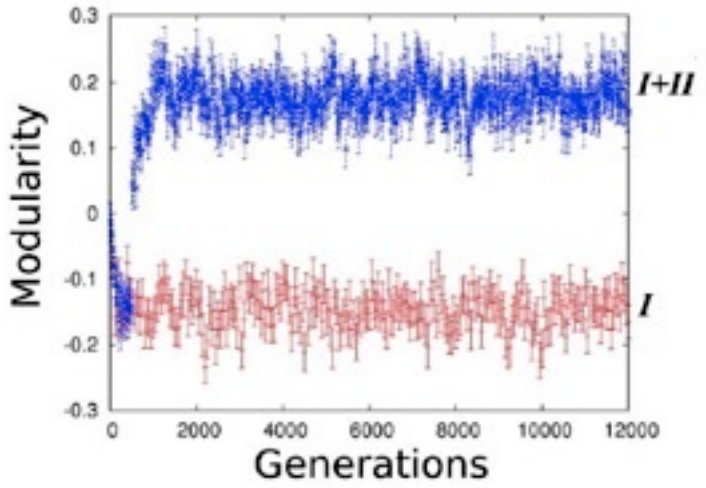


Fig. 29. Aumento di modularità non transitorio per la selezione del pattern II. La modularità delle reti migliori raggiunge un livello che si mantiene per almeno 10.000 generazioni per raggiungere entrambi i pattern. Tale livello è superiore a quello di reti selezionate per raggiungere un solo pattern I. La lunghezza delle linee rappresenta un errore standard. Il grafico mostra i risultati per 100 popolazioni in evoluzione su ogni regime di selezione.

I risultati ottenuti sono indipendenti da variazioni di pattern e parametri, infatti se si diminuisce il rate di mutazione e anche se il tempo necessario per evolvere I e II poi aumenta, la modularità continua ad aumentare significativamente. La modularità aumenta ulteriormente anche quando è già aumentata. E' stato chiesto se tali osservazioni fossero sensibili alla supposizione che i pattern del gene individuale contribuissero alla fitness aggiuntiva. Modificando tale ipotesi si ha ancora un aumento significativo di modularità.
Inoltre, l'aumento della modularità si riscontra anche per reti con più geni per cui tale comportamento non dipende dal numero di geni.
In una successiva analisi è stato chiesto se l'aumento di modularità dipende dall'identità degli stati di attività I e II trovando che alcuni geni hanno stesso stato di attività nei due pattern. Per esempio, la modularità aumenta anche quando i pattern differiscono nell'attività di entrambi i geni 3 o 7. Aumenta quando entrambi sono scelti a caso, eccetto per coppie con meno di due stati di attività diversi, questi vengono scartati. Non aumenta quando tutti i geni negli stati I e II differiscono nella loro espressione. Tale risultato non dovuto alla mancanza di adattamento, dal momento che le reti in grado di raggiungere entrambi i pattern sorgono in tutte le popolazioni che evolvono. Nel loro insieme, queste osservazioni mostrano che la modularità non aumenta solo per un pattern specifico, si tratta di una risposta generica dell'evoluzione. La distinzione fra i due insiemi di geni, quelli con identica e quelli con diversa attività in entrambi i pattern, è essenziale per l'evoluzione della modularità. La modularità aumenta solo in questo caso, i moduli sorgono per diminuire gli effetti dei geni con attività immutabile su geni con cambiamento di espressione in I e II e viceversa.
In caso affermativo, i moduli devono corrispondere a insiemi di geni richiesti per spostare la propria attività.

5.1.2

La modularità divide le reti per stato di attività unico e condiviso

Stabilito che l'evoluzione della modularità richieda geni con entrambi stati di attività diversi e condivisi, è stato chiesto se la partizione dei moduli è congruente con questi due insieme di geni.
Un modulo non tende a includere geni con diversi stati condivisi, mentre un altro include geni con stati diversi in I e II?
Si evolvono 300 popolazioni di rete, prima verso il pattern I e dopo verso entrambi I e II. Nel corso dell'evoluzione è determinata, per una delle migliori reti in ciascuna popolazione, la frequenza di p_{s-s} in cui due geni con stati condivisi in I e II sono nello stesso modulo, la frequenza p_{n-n} in cui due geni con stati diversi in I e II sono nello stesso modulo e la frequenza p_{s-n} con cui un gene specifico con stato condiviso e un gene con stato non condiviso sono nello stesso modulo (Fig. 30A). Selezionando i pattern I e II, p_{s-s} e p_{n-n} aumentano, p_{s-n} diminuisce (Fig. 30B).

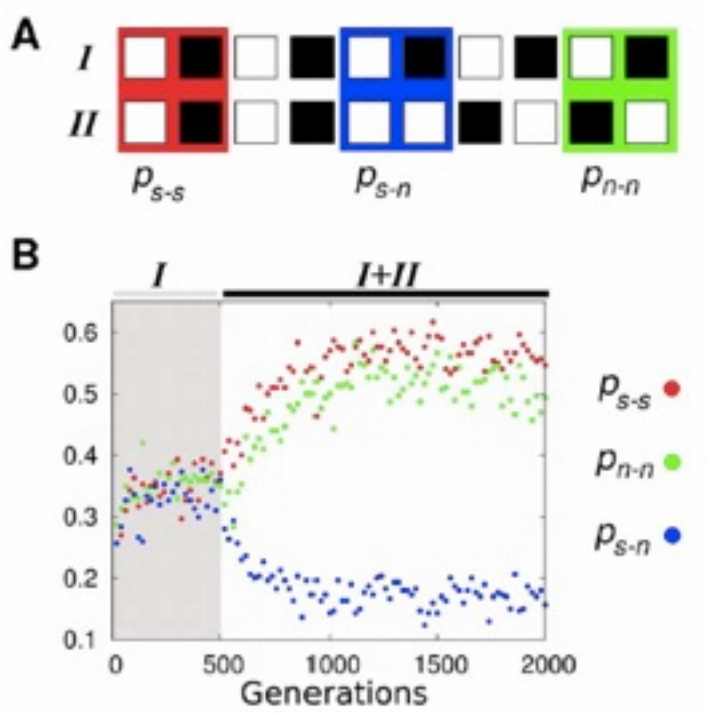


Fig. 30. Reti divise in base ai geni con stati di attività condivisa e non.

I geni con stati che cambiano notevolmente da una parte all'altra tutti i pattern selezionati, condivisi e non, tenderanno a essere inclusi nello stesso modulo e verranno tenuti separati dagli altri geni. Questo è esemplificato (Fig. 28D, E) confrontando una delle reti ottimali dopo aver selezionato il pattern I con una delle reti ottimali dopo la selezione per entrambi. Quest'ultimo è suddiviso in moduli in cui geni con stati distinti e condivisi dei pattern I e II sono a parte.

5.1.3

La modularità aumenta con la selezione di un terzo pattern

La modularità nasce solo dove la selezione favorisce il raggiungimento di due pattern o aumenta ancora con più pattern?
Si analizzano 100 popolazioni in evoluzione in cui la prima selezione favorisce un pattern I (500 generazioni), un pattern II (I + II, successive 1.500 generazioni) e un pattern aggiuntivo III (I + II + III, ultime 3.000 generazioni). I pattern condividono l'attività di alcuni geni e si differenziano in altri. Appena la selezione per il terzo pattern inizia, sorgono sempre più piccoli gruppi di geni la cui attività cambia (Fig. 31A).

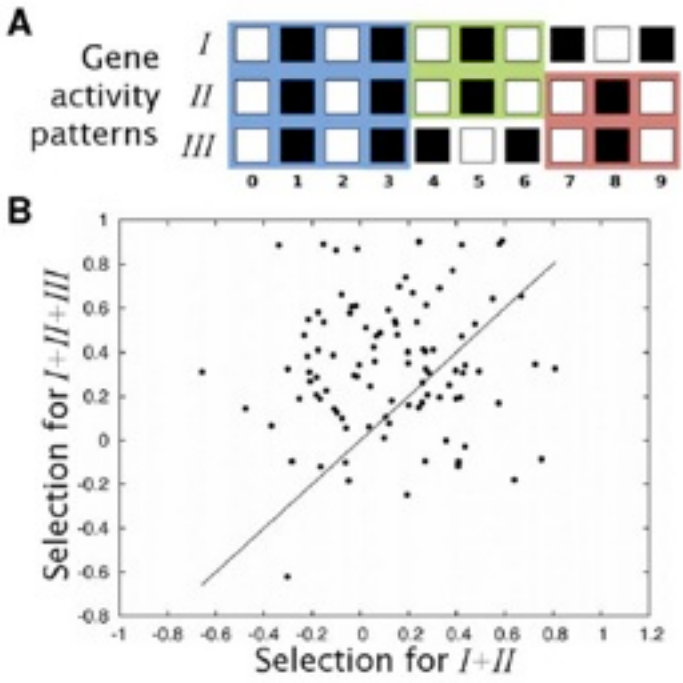


Fig. 31. La modularità aumenta ulteriormente dopo la selezione di un terzo pattern III. **(A)** Pattern I, II e III. Quadrati bianchi: geni attivi. Neri: geni inattivi. I colori di sfondo distinguono i geni che cambiano il loro stato di attività in tutti i pattern selezionati. In questo caso, l'ulteriore pattern causa piccoli gruppi di geni la cui attività cambia. **(B)** Asse orizzontale: la modularità delle reti più adatte dopo la selezione I e II. Asse verticale: la modularità delle reti più adatte dopo altre 3000 generazioni di selezione per I e II.

Le interazioni fra gruppi potrebbero ostacolare l'evoluzione dell'adattamento per cui devono essere contro selezionate determinando un ulteriore aumento di modularità. Dopo la selezione per i pattern I e II, si osserva un significativo aumento in modularità, aumenta ancora dopo la selezione del pattern III (Fig. 31B). Inoltre, è osservato un numero di moduli aumentato in reti con alta fitness dopo la selezione dei pattern I e II. Questo numero aumenta ulteriormente dopo la selezione dei pattern I, II e III. L'aumento di modularità dopo la selezione dei tre pattern avviene a causa della comparsa di nuovi moduli e non è una conseguenza del miglioramento di precedenti moduli evoluti.
E' analizzata anche come la probabilità di due geni dello stesso modulo cambi in tutta l'evoluzione, trovando che la frequenza di due geni è nello stesso modulo nelle migliori reti di ogni popolazione in evoluzione ma cambia a seconda che tali geni modifichino la loro attività attraverso pattern selezionati. Per esempio (Fig. 31) l'attività dei geni 5 e 6 cambia notevolmente attraverso tutti i pattern: se in un pattern il gene 5 è attivo, allora il 6 è inattivo in quello stesso pattern e viceversa.
La frequenza con cui quei geni giacciono nello stesso modulo aumenta con l'evoluzione. Al contrario, l'attività dei geni 0 e 6 cambia notevolmente quando vengono selezionati i pattern I e II, ma non anche per il pattern III. La probabilità di quei geni nello stesso modulo aumenta prima della

selezione del pattern III. Dopo che la selezione del pattern III inizia, la probabilità che i geni 0 e 6 si trovino nello stesso modulo diminuisce.
Questi risultati dimostrano che i moduli che sorgono dopo la selezione del terzo pattern tendono anche a coincidere con insiemi di geni i cui stati cambiano notevolmente i pattern selezionati.
Il costo computazionale non ha permesso di esplorare ulteriori aumenti di modularità attraverso la selezione di pattern aggiuntivi.
La modularità aumenta finché c'è un aumento nel numero di gruppi di geni per i quali si favoriscono notevoli cambiamenti di attività.

5.1.4

La modularità facilita co-opzione

La modularità può aumentare la capacità di evolvere facilitando co-opzione, ossia la combinazione di moduli precedentemente evoluti per svolgere nuove funzioni?
Sono state selezionate reti per la loro capacità di raggiungere stabilmente tre pattern I, II e III (Fig. 32A). E' stata scelta la combinazione specifica dei pattern (Fig. 32A) in quanto: promuove l'evoluzione di un modulo compresi i geni 0-4 e un altro modulo compresi i geni 5-9; permette l'inserimento di un ulteriore pattern IV fatto interamente da stati di attività associati a moduli evoluti precedentemente (Fig. 32A, B).

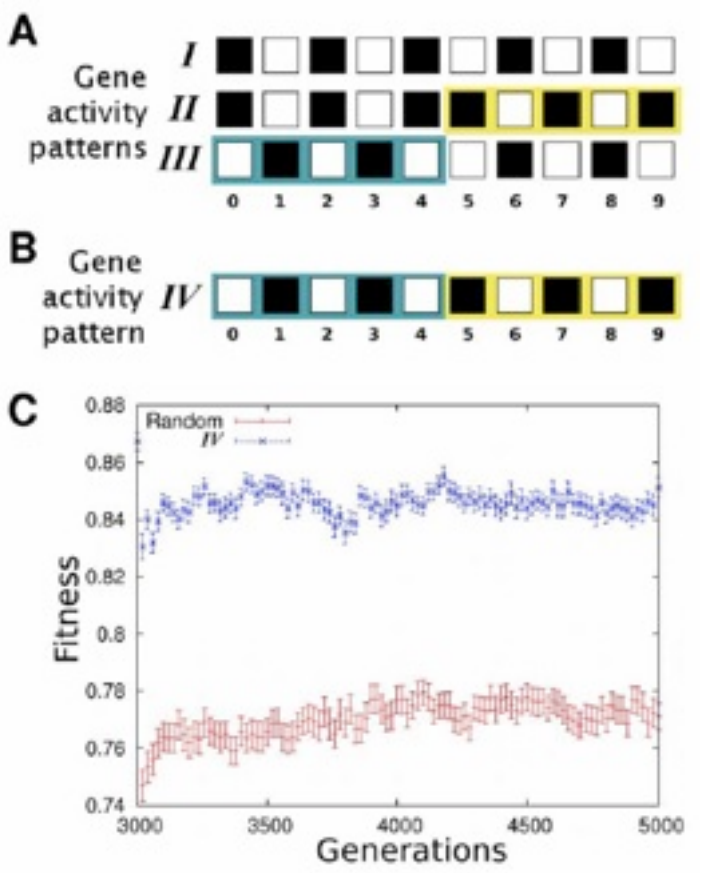


Fig. 32. Aumento rapido della fitness massima con co-opzione. **(A)** Reti raggiungono pattern I, II e III dopo 3.000 generazioni. Il regime di selezione promuove l'evoluzione di moduli contenti geni 0-4 da un lato e geni 5-9 dall'altro. **(B)** Dopo 3.000 generazioni la selezione favorisce il pattern IV (combinazione di pattern corrispondenti a quelli dei moduli evoluti precedentemente, come indicato dai colori dello sfondo). **(C)** Reti selezionate per raggiungere un quarto pattern aumentano la fitness più velocemente se questo pattern è il IV più di un pattern scelto a caso.

Dopo 3.000 generazioni, sono state sottoposte le reti in 100 popolazioni in evoluzione per selezione favorendo così un ulteriore pattern IV (Fig. 32B). Questo pattern condivide gli stati dei geni 0-4 con il III e lo stato dei geni 5-9 con il II, così il pattern IV può evolvere combinando moduli evoluti precedentemente in una nuova maniera. Approccio ripetuto in 100 popolazioni "*control*" dove il quarto pattern favorito è stato scelto casualmente con probabilità uguale per geni attivi e inattivi.
Non ci si aspetta che la selezione del pattern IV aumenti la modularità, poiché non causa un aumento nel numero di gruppi di geni con variazioni di attività notevoli. Piuttosto si ipotizza che la modularità faciliti l'acquisizione dell'evoluzione di un tale pattern rispetto ad altri.
Le reti con fitness alta sorgono molto più rapidamente quando IV è il nuovo pattern. Questo indica che tale pattern è molto più facile da raggiungere di un pattern casuale in popolazioni che sono state precedentemente selezionate per la loro capacità di raggiungere I, II e III (Fig. 32C). Gli stessi andamenti si verificano quando non solo sono considerate reti con più alta fitness ma anche quando si analizza la fitness media della popolazione. La selezione favorisce il raggiungimento del pattern IV nella stessa misura cosi come un qualsiasi pattern casuale in popolazioni di controllo.
L'aumento di fitness dipende dalla facilità con cui i nuovi pattern sono costruiti: è più facile evolvere pattern che combinano stati di attività di moduli precedentemente evoluti.
Si favoriscono reti con poche interazioni fra geni con uno stato immutabile e geni che adottano nuove funzioni. I geni con stati di attività correlati vengono a trovarsi nello stesso modulo e la modularità aumenta finché la selezione favorisce nuovi pattern che coinvolgono gruppi di geni sempre più piccoli la cui attività cambia notevolmente.
L'ipotesi alternativa per l'evoluzione della modularità citata all'inizio è lo scenario MVG che richiede popolazioni esposte a goal che evolvono e che cambiano nel tempo in modo tale che la modularità possa nascere ed essere mantenuta. Contrariamente alla specializzazione dell'attività del gene dove nuovi pattern devono essere raggiunti mentre i vecchi devono essere conservati, lo scenario MVG richiede cambiamenti per l'evoluzione dell'adattamento dopo che un goal cambia.
Al contrario il meccanismo con specializzazione richiede un genotipo per produrre pattern diversi in condizioni diverse e la modularità nasce per evitare di impedire di raggiungere pattern selezionati diversi in uno stesso genotipo.
La specializzazione può quindi essere più adatta se si hanno esigenze ambientali che non variano costantemente.

5.2

Specializzazione funzionale da duplicazione di moduli

La duplicazione di unità funzionali porta a specializzazione dell'evoluzione di unità duplicate. Contrariamente all'evoluzione di unità non derivanti da duplicazione, i task funzionali tendono a essere distribuiti fra unità ridondanti senza alcuna divisione di funzione.
Perché e come la duplicazione porta a specializzazione funzionale?
Perché l'evoluzione di unità ridondanti cablate nel sistema non nate da duplicazione, non portano a specializzazione?
Sono state condotte diverse simulazioni [CNPW00] confrontando:

- una semplice rete neurale feed-forward non modulare;
- un'architettura modulare cablata (progettata con modularità all'inizio che rimane fissa durante l'intera evoluzione);
- un'architettura modulare da duplicazione (si evolve da una popolazione di reti non modulari come risultato della duplicazione).

E' stata usata una probabilità di mutazione 1% (2% dei bit del genotipo selezionato a caso sostituito da un nuovo valore) in 10 simulazioni di ciascuna delle 3 diverse architetture.
Ogni simulazione inizia con popolazioni di 100 reti con pesi casuali e per 1000 generazioni.
Le architetture modulari originate da duplicazione favoriscono l'emergere di specializzazione di modulo funzionale?
L'architettura modulare cablata fornisce risultati migliori (Fig. 33) dell'architettura non modulare (prestazione media e picco del miglior individuo in ogni generazione). Da architettura non modulare a architettura modulare cablata si nota minore potenza di calcolo, ossia numero di neuroni e connessioni.
In entrambe, le condizioni il livello di prestazione aumenta fino a raggiungere un livello alto. Le popolazioni con moduli raggiungono un livello di prestazione superiore e in meno tempo (meno generazioni). Più precisamente, dopo un centinaio di generazioni di sovrapposizione di prestazione nelle due condizioni, popolazioni con moduli iniziano a essere superiori a popolazioni senza moduli e la modularità aumenta e rimane fino alla fine dell'evoluzione. Più evidente se si considera la prestazione dell'individuo migliore.
L'ipotesi di modularità implicita nel realizzare tale risultato può essere testata indirettamente variando il rate di duplicazione. Media e picco diminuiscono linearmente con run ate di duplicazione basso.

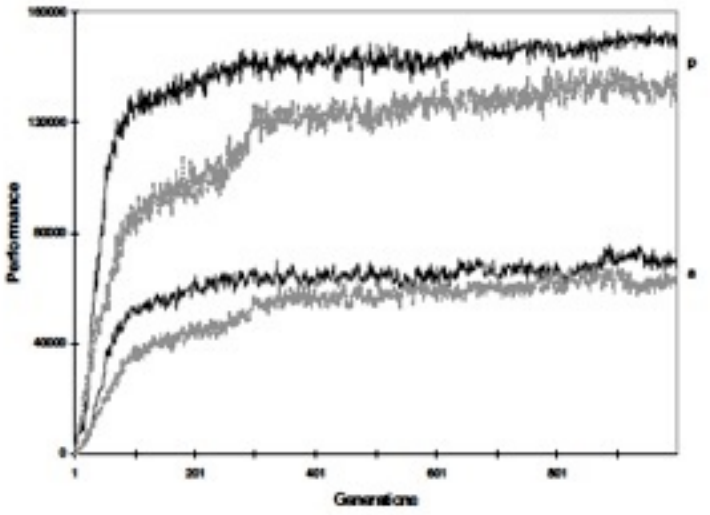


Fig. 33. Media (a) e picco di prestazione (p) di una popolazione con architettura non modulare (curva grigia) e di una popolazione con architettura modulare cablata (curva nera). Media di 10 esecuzioni diverse.

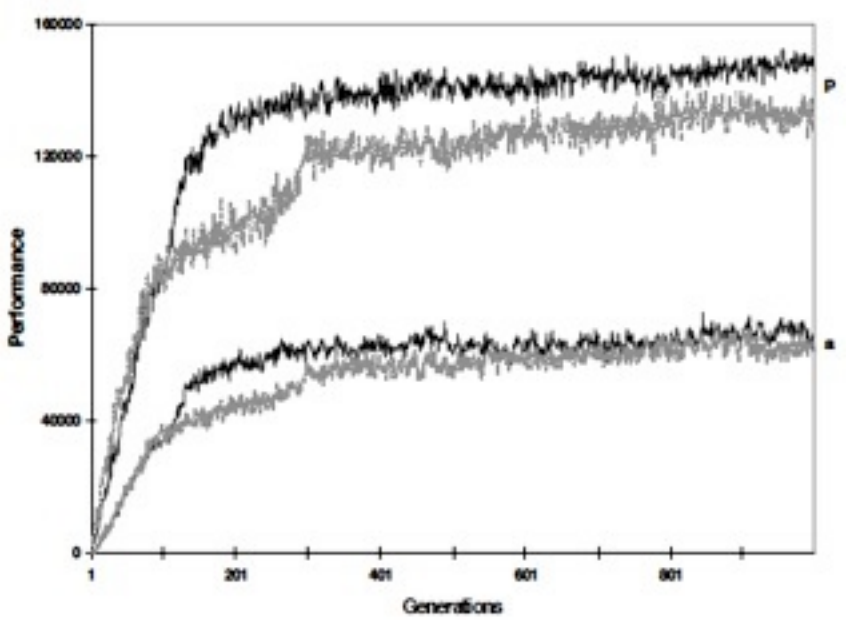


Fig. 34. Media e picco di popolazioni con architettura non modulare (curva grigia) e di popolazioni con modularità da duplicazione (curva nera) con rate di duplicazione 0.04%. Media di 10 esecuzioni diverse.

La Fig. 35 mostra i risultati ottenuti con architettura modulare da duplicazione con un rate di duplicazione 0.02% e la confronta con un architettura non modulare: il vantaggio della modularità è perso dimostrando l'importanza dell'interazione fra mutazione e rate.

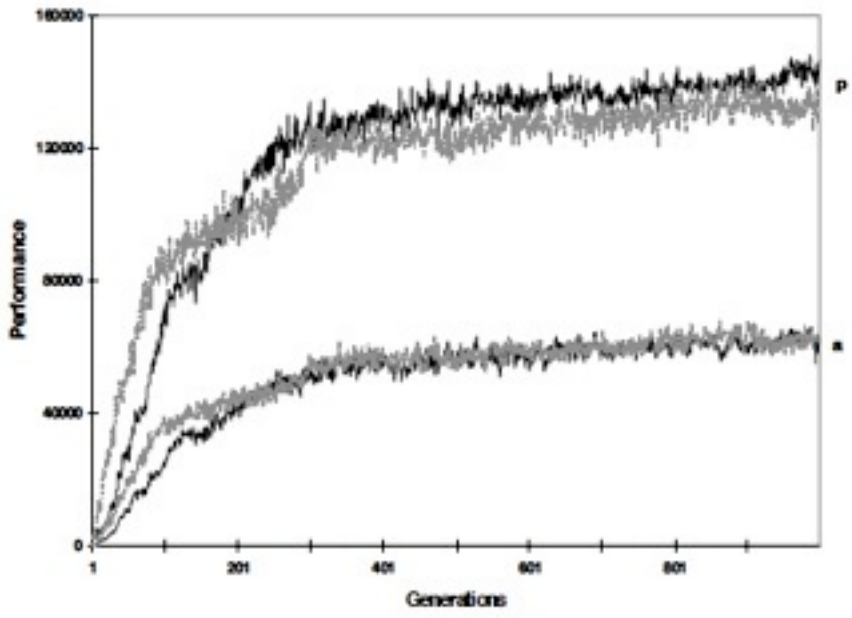


Fig. 35. Media e picco di prestazione di popolazioni con architettura non modulare (curva grigia) e di popolazioni con architettura modulare da duplicazione (curva nera) con rate di duplicazione 0.02%. Media di 10 esecuzioni diverse.

Confrontando le prestazioni di un'architettura modulare cablata e una da duplicazione, le popolazioni non differiscono in termini di prestazioni generali eccetto per la crescita della prestazione che è leggermente più lenta nella popolazione con moduli da duplicazione (Fig. 36). Questa differenza si spiega osservando che nell'architettura modulare da duplicazione, alcune generazioni devono prima vedere se la duplicazione possa avvenire e poi i moduli duplicati possono distinguersi fra loro.

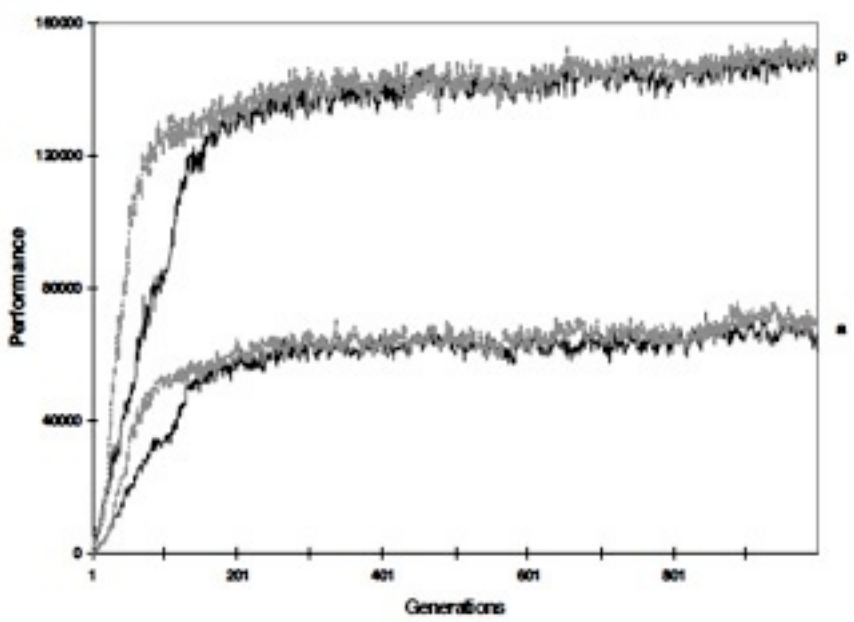


Fig. 36. Media (a) e picco di prestazione (p) della popolazione con architettura modulare cablata (curva grigia) e della popolazione con architettura modulare da duplicazione (curva nera) con rate di duplicazione 0.04%. Media di 10 esecuzioni diverse.

Non esiste corrispondenza uno-a-uno fra moduli interni e vari sottocomportamenti, ciò non implica che tutti i moduli interni contribuiscano ai diversi sotto comportamenti allo stesso modo.
Sebbene i risultati dimostrino che la modularità è utile nel produrre diversi comportamenti complessi, non necessariamente si trova una corrispondenza diretta uno-a-uno fra moduli e sotto comportamenti più semplici. Quindi non tutti i moduli interni contribuiscono a tutti i diversi sottocomportamenti nello stesso modo. Sebbene ogni modulo possa contribuire alla produzione di diversi comportamenti complessivi, un singolo modulo o un gruppo di moduli può essere coinvolto in uno solo o in un paio di sottocomportamenti e cioè i moduli possono avere un certo livello di specializzazione. Per illustrare ciò, si considera la Fig. 37. Sebbene le entità fenotipiche P1, P2, P3 contribuiscano tutte a produrre un sottocomportamento B1, P2 ha responsabilità principale, P1 e P3 contribuiscono meno. Similmente P1 e P3 hanno responsabilità principale su B2. Questo tipo di specializzazione è un vantaggio da un punto di vista dell'evoluzione se diversi sottocomportamenti hanno funzioni diverse (se un singolo sottocomportamento o un insieme di sottocomportamenti sono i principali responsabili di una singola funzione di adattamento).

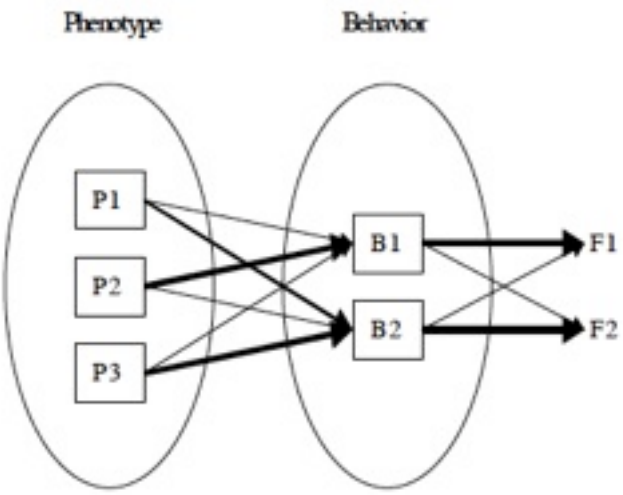


Fig. 37. Sinistra: organizzazione di un sistema a livello di fenotipo (P1, P2, P3 rappresentano diversi sottocomponenti del fenotipo). Centro: organizzazione del comportamento corrispondente (B1 e B2: due diversi sottocomportamenti). Destra: funzioni del comportamento complessivo. La direzione delle frecce indica l'importanza di un'entità nel determinare un'altra entità.

Nell'architettura modulare da duplicazione, i moduli sembrano essere più specializzati in sottocomportamenti specifici, meno vero nell'architettura modulare cablata. Questi risultati sembrano sostenere il modello proposto da Hughes (1994) e Ohno (1970), i quali assumono che tale specializzazione potrebbe sorgere quando i geni che servono funzioni multiple sono duplicati.
Dopo la duplicazione, i geni sono rilasciati da richieste in conflitto e ogni copia può specializzarsi in una delle diverse funzioni del gene di partenza. La duplicazione del gene è soltanto uno dei fattori che può portare a specializzazione funzionale (Wagner & Altenberg, 1996).
L'evoluzione può portare a un certo livello di specializzazione sotto determinate condizioni. Per quanto riguarda le prestazioni generali non ci sono differenze significative dopo 1000 generazioni fra individui con architettura da duplicazione e individui con architettura modulare cablata. La specializzazione funzionale di moduli interni non comporta maggiore capacità di adattamento. Risultato in contraddizione con l'assunzione che individui con strutture interne specializzate hanno un alto livello di evoluzione (maggiore probabilità di ottenere un miglioramento attraverso variazioni casuali). Il fatto che le due classi di individui raggiungano quasi lo stesso livello di prestazione può essere spiegato dal fatto che individui non specializzati siano già vicini a prestazioni ottimali.
Per comprendere i meccanismi attraverso i quali la duplicazione di unità strutturali favorisce la specializzazione di modulo è eseguita un'analisi con un individuo migliore dell'ultima generazione e si risale agli antenati fino alla prima generazione. Cosi l'intera discendenza dell'individuo migliore dell'ultima generazione è ricostruita. Simulazione condotta (Calabretta e al. 1998) con una discendenza di 1000 individui, uno per generazione. L'individuo rappresentante la discendenza vincente nella prima generazione ha un'architettura non modulare e i pesi casuali. Gli individui della generazione vincente delle generazioni successive cambiano progressivamente l'architettura per duplicazione e mutazione.

5.2.1

Risultati

Architetture modulari cablate e da duplicazione forniscono risultati migliori in termini di velocità di evoluzione e livello di stato stazionario raggiunto alla fine delle prestazioni, rispetto ad architetture non modulari. In termini di prestazioni complessive non c'è differenza.
Le architetture modulari originate da duplicazione tendono a favorire una specializzazione funzionale del modulo rispetto ad architetture cablate. Spiegazione trovata per gli effetti delle mutazioni. In entrambe i casi, le mutazioni possono avvenire sulla parte che regola la struttura (controllando il modulo che determina l'output della rete in risposta ad alcuni input) o sulla parte strutturale (controllando il tipo di uscita che il modulo genera in risposta all'ingresso).
Nell'architettura da duplicazione, una mutazione può anche produrre duplicazione di modulo. Inizialmente, la duplicazione non produce alcuna conseguenza sull'adattamento perché i due moduli duplicati sono perfettamente identici. Successivamente una nuova mutazione che avviene sulla parte che regola, può fornire un modulo di controllo dell'output per alcuni input e l'altro modulo di controllo dell'output per altri ingressi. Una mutazione sulla parte strutturale può portare a specializzazione e quindi a conseguenti capacità di adattamento.
Nell'architettura da duplicazione, la commutazione fra moduli concorrenti è inizialmente neutrale da un punto di vista di fenotipo e crea le condizioni favorevoli per un utilizzo continuo dei due moduli, una ritenzione selettiva di una mutazione favorevole che avviene sulla parte strutturale e una maggiore probabilità di specializzazione.
La duplicazione di moduli con adattamento facilita in parte l'evoluzione della specializzazione funzionale e comprende prima l'acquisizione di un cambiamento per regolare i moduli duplicati e poi l'adattamento dei moduli al nuovo contesto funzionale.
Questo meccanismo porta a un co-adattamento fra parti, funzionali e normative che bloccano il sistema in uno stato con specializzazione funzionale.

CAPITOLO 6

Simulare l'evoluzione di sistemi neurali modulari *

Due task indipendenti sono eseguiti in maniera più efficiente separatamente da due moduli dedicati invece che da un sistema omogeneo (completamente distribuito), questo per risolvere il problema dell'interferenza.
Per apprendere più task, la struttura di base è quella di reti feed-forward, con tre livelli di neuroni semplificati (Fig. 38). Le attivazioni dei layer di input rappresentano l'input del sistema e passano mediante connessioni con pesi al layer nascosto, dove ogni unità somma i suoi input e passa il risultato attraverso qualche funzione (per es. una sigmoide) per produrre il proprio livello di attivazione. Le attivazioni poi passano da un secondo livello di connessioni con pesi al layer di output dove sono nuovamente sommate per produrre le attivazioni di output (ad es. le rappresentazioni "What-Where").
I pesi delle connessioni sono appresi da un algoritmo di apprendimento con discesa a gradiente e modificati iterativamente in modo che la rete produca sempre output accurati. In tale contesto, la modularità si riferisce alla connettività fra layer di output e nascosti.

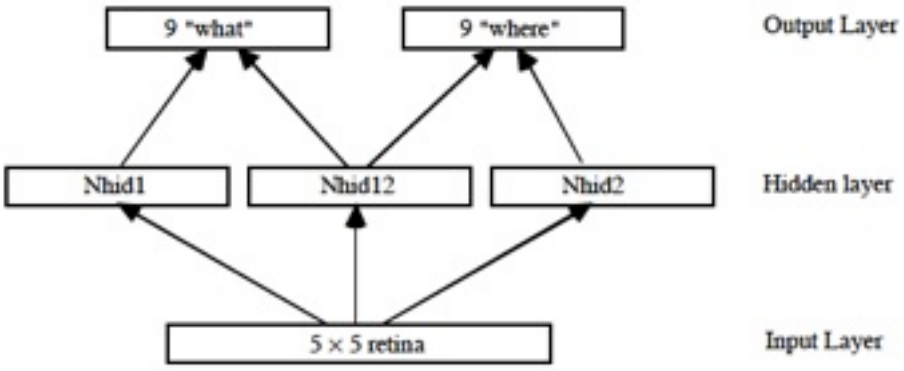


Fig. 38. Architettura del modello base di rete neurale per i task "What-Where".

Durante l'apprendimento un'unità nascosta usata per elaborare informazione per due o più unità di output potrebbe ricevere contributi di aggiornamento di pesi in conflitto con conseguente degrado delle prestazioni rispetto a una rete con un insieme separato di unità nascoste per ciascuna unità di output (Plaut & Hinton, 1987).
La modularità con un insieme di unità nascoste (o modulo) per ogni unità di output è inefficiente in termini di risorse di calcolo. Un algoritmo di apprendimento efficiente deve essere in grado di trattare adeguatamente gli aggiornamenti di pesi in conflitto.

* J.A. Bullinaria "Simulating the Evolution of Modular Neural Systems", in Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society, 2001.
J.A. Bullinaria *"Understanding the Advantages of Modularity in Neural Systems"*, in Cognitive Science, N.31, 2007.

Una possibile soluzione efficiente potrebbe essere quella di suddividere le unità nascoste in insiemi disgiunti corrispondenti a task di output distinti. Quando si usano algoritmi di apprendimento standard, l'elaborazione a livello nascosto tende a diventare completamente distribuita e non c'è emergenza spontanea di modularità (Plaut, 1995; Bullinaria, 1997).
Altra ipotesi [JJO92] è che in sistemi opportunamente semplificati con connessioni corte si possono apprendere architetture modulari.

6.1

Applicazione al modello "What-Where"

E' stato studiato lo stesso modello di rete neurale semplificata usato da Rueckl e al. (1989). Il task consiste in un mapping dell'immagine retinica semplificata (matrice 5 x 5) su una rappresentazione semplificata del "What" (vettore binario a 9 bit con un bit "on") e del "Where" (vettore binario a 9 bit con un bit "on"). Sono usati gli stessi 9 pattern di input e 9 posizioni, come negli studi precedenti dando gli stessi 81 input per l'apprendimento. Ciascuno dei 9 pattern consiste di un insieme diverso di 5 celle "on" su una matrice 3 x 3 e le 9 posizioni corrispondono ai possibili centri di un array 3 x 3 in un array completo 5 x 5.
La Fig. 38 mostra la rete di base studiata (Rueckl e al.) con 25 unità di input, 18 di output e 81 pattern di apprendimento. Le linee con frecce rappresentano una connettività completa e *Nhid1, Nhid12, Nhid2* specificano quante unità nascoste ci sono in ogni blocco. Rueckl e al. (1989) hanno studiato la rete completamente distribuita: (*Nhid1 = Nhid2* = 0) e la rete puramente modulare (*Nhid12* = 0).
Se il numero massimo di unità nascoste *Nhid = Nhid1 + Nhid12 + Nhid2* è fisso allora bisogna definire solo due architetture innate di parametri: *Con1 = Nhid1 + Nhid12* e *Con2 = Nhid2 + Nhid12* corrispondenti al numero di unità nascoste connesse a ogni blocco di output.
Per simulare il processo evolutivo è presa un' intera popolazione di singole istanze di ogni modello ed è consentito loro di imparare, procrearsi, morire in maniera simile ai sistemi reali. Il genotipo di ciascun individuo dipende da quelli dei genitori e contiene tutti i parametri innati adatti. Durante la propria vita imparerà dall'ambiente come meglio modificare i pesi per eseguire nel modo più efficace. Ogni individuo morirà dopo la riproduzione dei figli. Si considera una buona approssimazione della relazione fra funzione di fitness del singolo task e sopravvivenza o procreazione della fitness. Invece di addestrare ciascun membro dell'intera popolazione per un tempo fisso e scegliere poi il più adatto a riprodursi e formare la successiva generazione, le popolazioni conterranno individui concorrenti di tutte le età, ognuno con possibilità di morire e procrearsi in ogni fase. Ogni individuo impara dalla propria esperienza con l'ambiente (ad es. un insieme di dati di apprendimento/testing) e ha la propria fitness determinata.
Un sottoinsieme casuale di individui meno adatti con un sottoinsieme casuale di individui più vecchi morirà e sostituito da bambini con un genitore scelto a caso dai membri più adatti della popolazione, la quale sceglie casualmente un compagno dal resto dell'intera popolazione. Ogni figlio eredita caratteristiche da entrambi i genitori. E' opportuno avere ogni individuo con pesi iniziali casuali e consentire all'architettura di evolversi.
I parametri ottimali di apprendimento varieranno con l'architettura e devono evolvere insieme con l'architettura. L'importante è fissare i parametri dell'evoluzione correttamente a seconda dei dettagli del problema.

6.2

Modello base

Il modello base è costituito dall'algoritmo di apprendimento con discesa a gradiente (Rueckl, 1989) tenuto fisso con *momentum* sulla funzione di costo *Sum-Squared Error E* (Hinton, 1989).
Gli output sono: 0.1 e 0.9 invece di 0 e 1 e gli output adatti presunti senza errore. Le reti imparano meglio rate di apprendimento diversi per ciascuno dei diversi layer di collegamento.
Per assicurare che tutti gli apprendimenti siano al loro potenziale completo, ogni rete ha cinque parametri di apprendimento: il rate di apprendimento η_{IH} per l'input al layer nascosto, η_{HB} per i layer nascosti e il parametro *momentum* α. Di seguito è riportata l'equazione standard di aggiornamento dei pesi:

$$\Delta w_{ij}(n) = -\eta_L \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(n-1)$$

Ogni genotipo ha due parametri per controllare l'architettura e cinque per controllare i rate.
La distribuzione di costo *Sum Squared Error* è distorta per tutta la popolazione cosi sono state definite delle fitness evolutive individuali a –log(Cost).
Bullinaria (2001) ha trovato che l'evoluzione può dipendere da condizioni iniziali, come la distribuzione dei parametri innati di tutta la popolazione iniziale e che la popolazione risulta in uno stato vicino ottimo se inizia con una distribuzione ampia dei rate di apprendimento invece di aspettare che le mutazioni portino il sistema da uno stato in cui c'è poco apprendimento a uno completo. I rate della popolazione iniziale sono scelti a caso nell'intervallo [0.0, 2.0] e i parametri *momentum* a caso nell'intervallo [0.0, 1.0].
Seguendo Rueckl e al. (1989) i pesi iniziali scelti a caso nell'intervallo [0.0, 0.3].
I parametri innati sono evoluti con 18 unità nascoste in totale (Fig. 39) e i parametri di apprendimento si stabilizzano presto. Dopo un inizio non modulare, la popolazione evolve rapidamente verso un architettura modulare con *Nhid12* prossima allo zero.

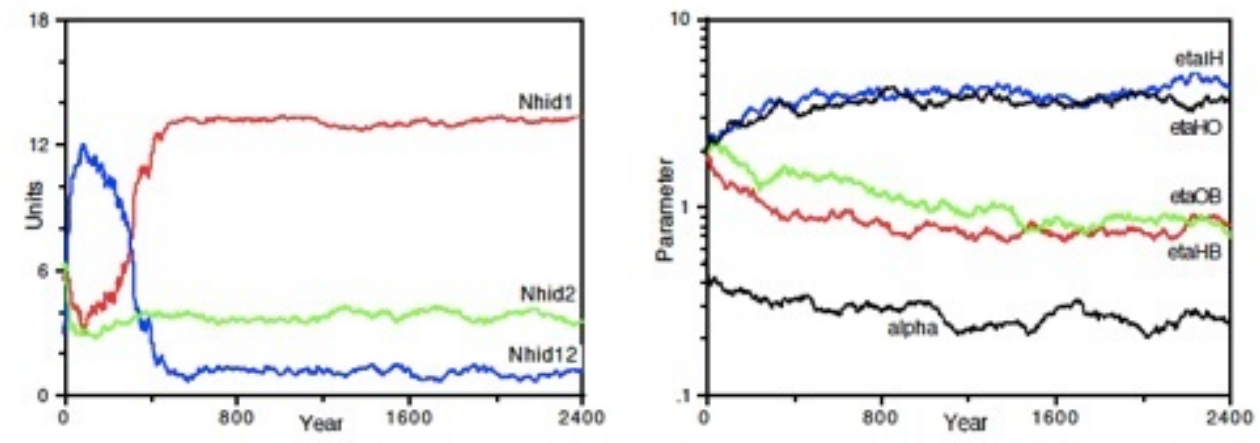


Fig. 39. Evoluzione del modello con funzione di costo Sum-Squared Error e funzione di fitness Log Cost.

CAPITOLO 7

Modularità da competizione [*]

Un caso ulteriore di modularità è quello di architetture connessionistiche. Le reti componenti una tale architettura competono per imparare dei pattern di apprendimento. Risultato della competizione è che reti diverse imparano pattern diversi e quindi imparano a calcolare funzioni diverse. L'architettura esegue scomposizione di task, impara a partizionare un task in due o più task funzionalmente indipendenti e assegna reti distinte per apprenderli. In più assegna anche a ciascun task della rete la topologia più adatta. Tale proprietà ha implicazioni sull'efficacia dell'apprendimento e sulla capacità di generalizzare dell'architettura modulare. Le architetture modulari hanno vantaggi in termini di: velocità di apprendimento, generalizzazione di funzionalità, rappresentazione e capacità di soddisfare vincoli.
L'architettura connessionistica modulare (Fig. 40) è costituita di due tipi di reti: *expert network* e *gating network*. Le prime competono per imparare i pattern training mentre la gating network media la competizione. Dopo l'apprendimento, le Expert Network 1, 2 calcolano funzioni differenti utili in regioni diverse dello spazio di ingresso. I vettori y_1 e y_2 indicano gli output. La Gating Network decide se le due Expert Network sono applicabili al momento. Gli scalari g_1 e g_2 indicano le due unità di output della gating network. L'output dell'intera architettura y è dato da $g_1 y_1 + g_2 y_2$.

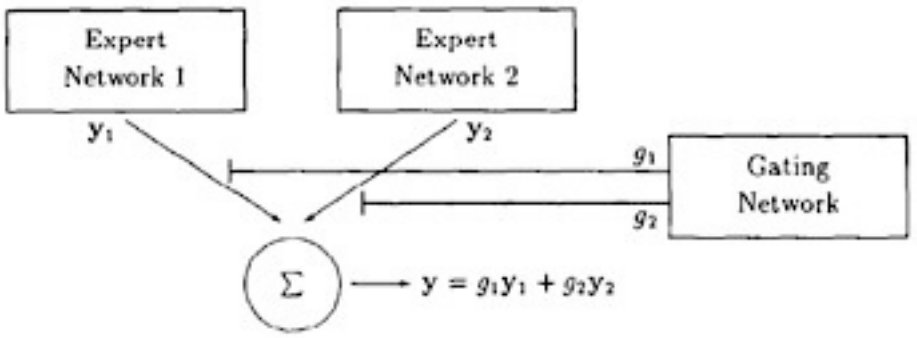


Fig. 40. Architettura connessionistica modulare

[*] R. A. Jacobs, M. I. Jordan, A. G. Barto "Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks" in Cognitive Science, N.15, 1991.
M. Husken, C. Igel, M. Toussaint "Task-dependent evolution of modularity in neural networks" in Connection Science, N.14, Cap.13, 2002.

Quando $g_1 = 1$ e $g_2 = 0$ la Expert Network 1 determina l'output dell'architettura mentre quando $g_1 = 0$ e $g_2 = 1$ è la Expert Network 2 a determinare l'output.
In generale l'architettura può avere un numero qualsiasi di expert network. In fase di apprendimento i pesi di tutte le reti sono modificati simultaneamente con il backpropagation. Le regole di apprendimento sono basate su una minimizzazione delle diverse funzioni di errore fra l'uscita del sistema y e quella desiderata y^*. Tale errore J_y è dato da:

$$J_{\mathbf{y}} = \frac{1}{2}(\mathbf{y}^* - \mathbf{y})^T(\mathbf{y}^* - \mathbf{y})$$

Per ogni pattern training, la expert network si avvicina per produrre l'uscita desiderata. Nella competizione fra reti, questa è chiamata "vincitrice" mentre tutte le altre "perdenti". Se in un dato pattern training le prestazioni del sistema migliorano, allora i pesi della gating network sono adatti per portare l'output corrispondente alla expert network vincente verso 1. Mentre gli output delle expert network perdenti verso 0. Se non c'è miglioramento, i pesi della gating network spostano le uscite verso un valore neutro. Le prestazioni del sistema sono espresse dalla seguente equazione:

$$\overline{J}_{\mathbf{y}}(t) = \alpha J_{\mathbf{y}}(t) + (1 - \alpha)\,\overline{J}_{\mathbf{y}}(t - 1)$$

Se t è lo step al tempo corrente, l'errore J_y è una misura della prestazione corrente. Le prestazioni precedenti del sistema si calcolano mediante una media esponenziale ponderata di J_y negli step temporali prima di t. Questo valore è calcolato iterativamente dall'equazione sopra, α determina il cambiamento dei valori precedenti.
Le variabili binarie λ_{WTA} (WTA "*winner-take-all*") e λ_{NT} (NT "*neutral*") indicano se le prestazioni del sistema sono migliorate significativamente. Se

$$J_{\mathbf{y}}(t) < \gamma \overline{J}_{\mathbf{y}}(t - 1)$$

allora $\lambda_{WTA} = 1$ e $\lambda_{NT} = 0$, altrimenti $\lambda_{WTA} = 0$ e $\lambda_{NT} = 1$. γ è un fattore moltiplicativo che determina quanto più basso deve essere l'errore corrente rispetto alla misura degli errori precedenti, affinchè le prestazioni del sistema siano considerate migliorate. Se la prestazione dell'architettura è notevolmente migliorata ($\lambda_{WTA} = 1$) viene determinata quale uscita della expert network è più vicina all'uscita desiderata. L'errore per la expert network i come somma dell'errore quadratico fra output della expert network y_i e uscita desiderata y^*:

$$J_{\mathbf{y}_i} = \frac{1}{2}(\mathbf{y}^* - \mathbf{y}_i)^T(\mathbf{y}^* - \mathbf{y}_i)$$

La expert network vincente è la rete con errore più piccolo. Se la expert network i è vincitrice, allora il valore desiderato dell'unità di output i-esima della gating network g_i^*, è impostato a 1. Se la expert network i è perdente, g_i^* è impostato a 0. Se la prestazione dell'architettura non è migliorata significativamente (λ_{NT}) allora i pesi della gating network sono modificati in modo che tutte le uscite della gating network vengano spostati verso un valore neutro. Tale valore è $1/n$ con n: numero di expert network. La funzione errore della gating network J_G:

$$J_G = \lambda_{WTA} \frac{1}{2} \sum_{i=1}^{n} (g_i^* - g_i)^2 +$$

$$\lambda_{WTA} \frac{1}{2} (1 - \sum_{i=1}^{n} g_i)^2 +$$

$$\lambda_{WTA} \sum_{i=1}^{n} g_i (1 - g_i) +$$

$$\lambda_{NT} \frac{1}{2} \sum_{i=1}^{n} (\frac{1}{n} - g_i)^2.$$

Grazie alla definizione di λ_{WTA} e λ_{NT} i primi tre termini contribuiscono all'errore quando la prestazione è migliorata significativamente mentre solo il quarto termine contribuisce all'errore quando non è migliorata. Il primo termine è la somma dell'errore quadratico fra uscite desiderate e reali della gating network. Il secondo termine prende il valore più piccolo quando le uscite della gating network danno come somma 1. Il terzo termine prende il valore più piccolo quando le uscite della gating network sono valutate in binario.
L'effetto di cambiare i pesi della gating network per ridurre il secondo e terzo termine è in risposta a ciascun pattern di input, un unità di output si avvicina a 1 e tutte le altre a 0. Il quarto termine è la somma degli errori quadratici fra il valore neutro e le uscite effettive della gating network.
La riduzione di questo termine che si verifica solo quando la prestazione non è migliorata porta tutte le uscite della gating network ad avvicinarsi al valore neutro *1/n*.
Queste equazioni implicano che esistono tre tipi di interazioni fra reti di un architettura modulare. Il primo tipo: la gating network determina quanto ciascuna expert network contribuisce all'uscita del sistema; il secondo tipo: la prestazione delle expert network determina le uscite desiderate della gating network; il terzo tipo: la gating network determina quanto ogni expert network impara ogni pattern training.
Il vettore di errore *backpropagated* della Expert Network 1 è : $g_1(y^* - y)$ mentre della Expert Network 2 è : $g_2 (y^* - y)$.
Oltre a determinare quanto ogni expert network contribuisca all'uscita dell'architettura, la gating network determina anche le grandezze dei vettori di errore e quindi determina quanto ogni expert network impara di ogni pattern training.

7.1

Applicazione ai task "What-Where"

I vantaggi computazionali di usare sistemi distinti per eseguire due task deriva da un confronto delle prestazioni di due sistemi connessionistici sui task What-Where (Rueckl e al., 1989). Il primo fatto di una rete singola per eseguire entrambi i task e il secondo costituito da due reti, una per ogni task. Il sistema a due reti apprende i task più rapidamente e sviluppa rappresentazioni facilmente più interpretabili di quello a rete singola.
Secondo Rueckl e al. (1989) questo deriva dal fatto che le unità nascoste del sistema a rete singola ricevono informazione inconsistente, poichè connesse alle unità di output per entrambi i task ed è influenzato da *crosstalk* spaziale. Viceversa le unità nascoste del sistema a due reti non ricevono informazione inconsistente, poiché connesse a unità di output per un solo task.
Quindi la cosa migliore per un tale sistema è eseguire i task What-Where in reti distinte.
Sono state studiate un insieme di simulazioni per esaminare il comportamento dell'architettura in presenza di crosstalk temporale e l'altro in presenza di crosstalk spaziale e riferite a tre diversi periodi di tempo. Ad ogni step temporale viene presentato un sistema con una coppia input-output; durante ogni epoca un sistema è presentato con ciascuna coppia nell'insieme di apprendimento esattamente una volta con un'esecuzione di 100 epoche. La misura della prestazione è la percentuale di coppie input-output che il sistema esegue correttamente in ogni epoca.
Una coppia è eseguita correttamente quando ogni unità di output del sistema ha un'attivazione maggiore di 0.6 quando l'attivazione desiderata è 1 e al di sotto di 0.4 quando quella desiderata è 0. I risultati prodotti su una media di 25 esecuzioni.
La crosstalk temporale si verifica quando un sistema è addestrato per eseguire diversi task in tempi diversi. Sono state addestrate architetture modulari e reti singole per eseguire i task e ad ogni step ciascun sistema è addestrato per eseguire uno solo dei task.
Per ricevere i 25 input corrispondenti agli ingressi della matrice, è stato aggiunto un input indicativo chiamato *task specifier* e sono presenti 162 coppie input-output diverse (9 oggetti x 9 posizioni x 2 task).
Due modalità di apprendimento, una che evita crosstalk temporale chiamata *random training* e un'altra che non lo evita chiamata *blocked training*.
Con *random training* ad ogni step si seleziona un pattern di input a caso e il task specifier secondo una distribuzione uniforme sull'insieme dei 162 possibili pattern di input. La risposta desiderata è l'identità dell'oggetto o posizione dipendente. Con *blocked training,* il task specifier cambia una volta sola in ogni epoca. All'inizio di ogni epoca, uno dei due task è selezionato a caso (con probabilità 0.5) e i pattern di input presentati durante i primi 81 step hanno il task specifier settato per indicare il task selezionato. I vettori di input presentati negli ultimi 81 step hanno il task specifier settato ad altri task. Per esempio in un epoca, si potrebbe richiedere a un sistema di eseguire il task What per 81 step consecutivi seguiti dal task Where per altri 81 step.
Ad ogni step di un epoca, l'oggetto e la posizione sono selezionati a caso e la risposta desiderata dipende dal task specifier. Il primo sistema addestrato ad eseguire i task è una rete singola addestrata con algoritmo backpropagation. E' una rete con 26 unità di input, 18 unità nascoste e 9 di output. Le unità di input codificano la matrice 5 x 5 e il task specifier.
In (Fig. 41) sono mostrate le curve di apprendimento: in asse orizzontale c'è il numero di epoche e in asse verticale la percentuale di coppie input-output eseguite correttamente.
Come risultati si ha che la rete ha appreso più velocemente con apprendimento casuale che bloccato (a Epoca 50, la differenza di prestazione con apprendimento casuale e bloccato è statisticamente significativa e può essere spiegata dall'influenza del crosstalk temporale nel caso di apprendimento bloccato. Il crosstalk temporale attenua il rate di apprendimento.

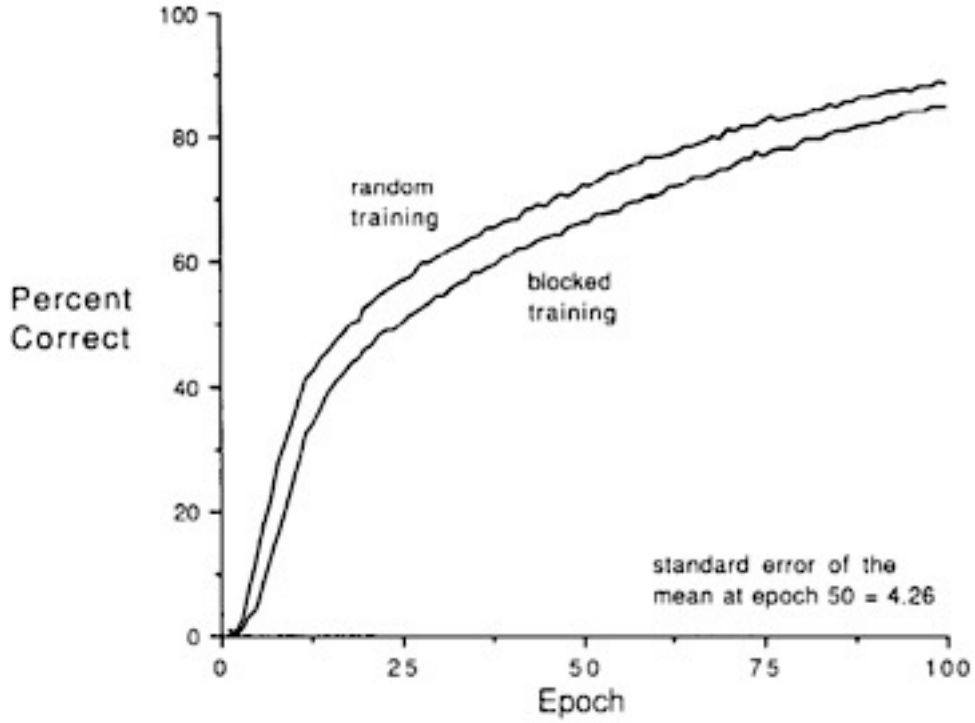


Fig. 41. Curve di apprendimento per la rete 26 → 18 → 9 sui task What-Where con *random* e *blocked training*.

Il secondo sistema appreso (Fig. 42) è identico al primo solo che ha 36 unità nascoste invece di 18. Per le prime 25 epoche, la rete ha appreso velocemente sia con blocked training sia con random training, poi da 25 a 85 epoche ha appreso molto più velocemente con random training (a Epoca 50, la differenza di prestazione fra random e blocked training è significativa.
In maniera simile al primo sistema, il crosstalk temporale attenua il rate di apprendimento e quindi un abbondanza di unità nascoste rende una rete più robusta alla presenza di tale crosstalk.

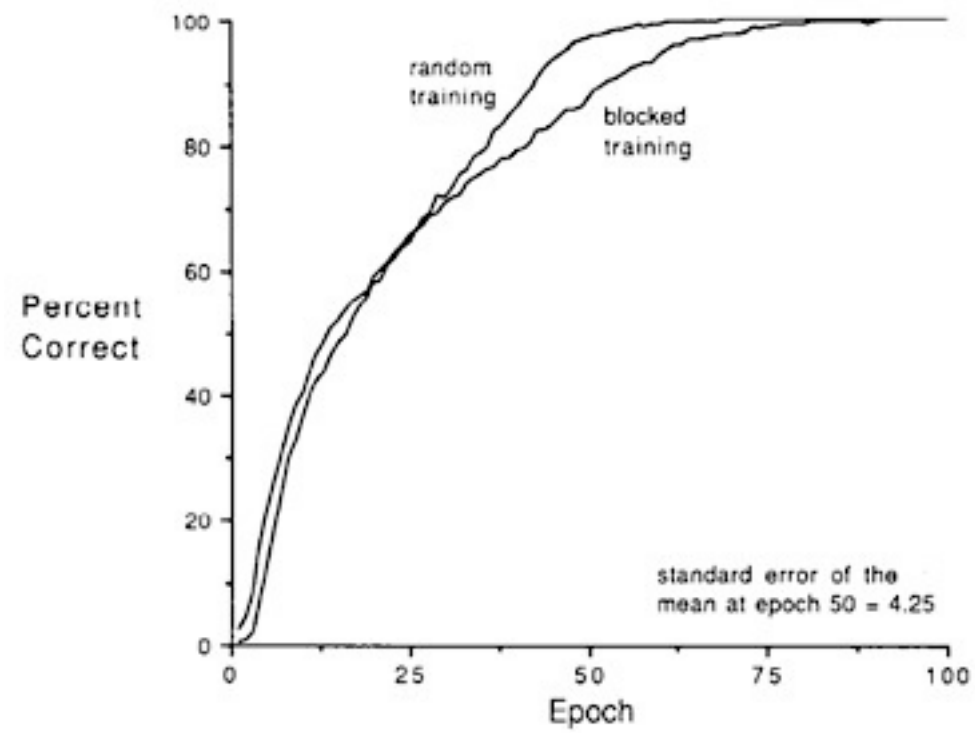


Fig. 42. Curve di apprendimento per la rete 26 → 36 → 9 sui task What-Where con *random* e *blocked training*.

L'architettura modulare applicata ai task (Fig. 43) consiste di tre *expert network* e una *gating network*. Ogni expert network ha 26 unità di input e 9 di output. Le unità di input codificano la matrice retina 5 x 5 e il task specifier. Due delle expert network sono reti multilayer con 36 e 18 unità nascoste rispettivamente. La terza expert network ha un singolo layer senza unità nascoste. La gating network ha un unità di input che codifica il task specifier e 3 di output corrispondenti alle tre expert network.

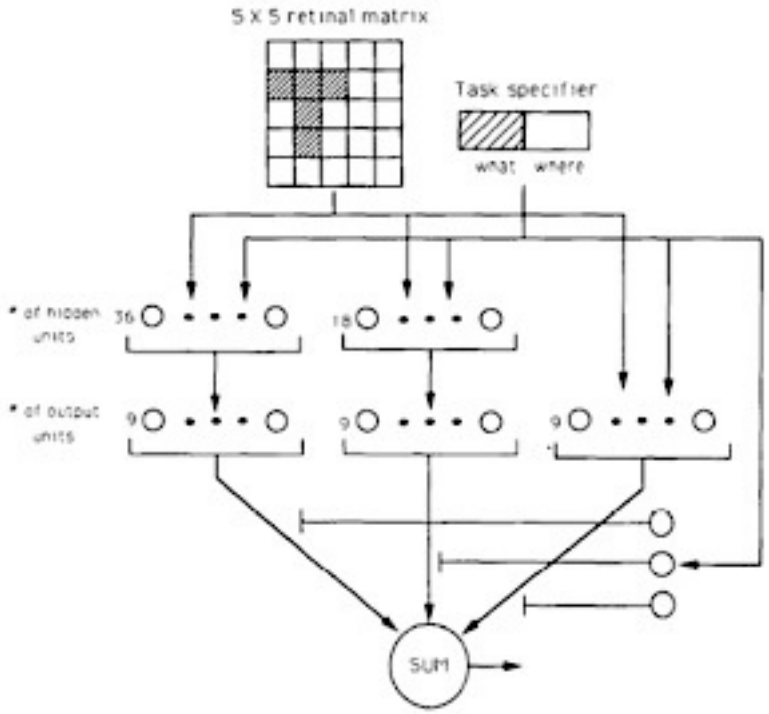


Fig. 43. Architettura Modulare simulata in esperimenti con crosstalk temporale.

Ci sono tre modi per tale architettura di imparare i task: una delle expert network multilayer potrebbe imparare a eseguire entrambi ma non è la soluzione migliore in termini di velocità di apprendimento e chiarezza della rappresentazione risultante; una seconda possibilità è che una delle expert network multilayer potrebbe imparare il task What e l'altra il Where. Questa soluzione indica che l'architettura modulare impara quello che è richiesto per eseguire i due task indipendenti e assegna reti distinte. La terza soluzione è la migliore dove una delle expert network multilayer ha imparato il task What e la expert network a singolo layer ha imparato il Where, mostrando cosi non solo una scomposizione dei task ma anche un assegnamento adeguato dei task alle expert network.
Almeno in alcune circostanze l'architettura modulare è capace di eseguire scomposizione di funzione e assegnare a ciascuna una rete con struttura adatta. Poiché esiste una piccola differenza di prestazione con i due tipi di apprendimento, tali risultati suggeriscono che l'architettura modulare è robusta in presenza di crosstalk temporale, dovuta alla capacità dell'architettura di assegnare reti distinte per imparare task diversi.
Un confronto delle curve di apprendimento (Fig. 44) mostra che entrambe le reti 26 → 36 → 9 e l'architettura modulare imparano i task più velocemente della rete 26 → 18 → 9.

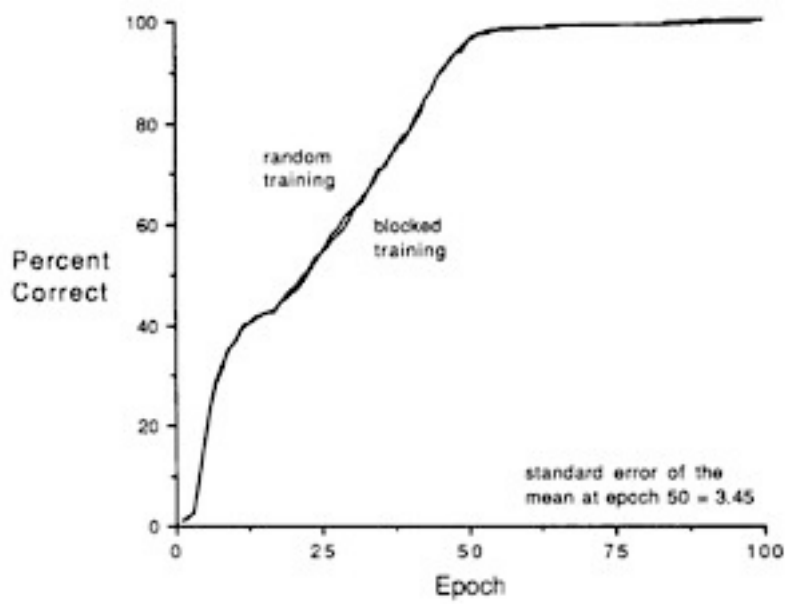


Fig. 44. Curve di apprendimento per l'architettura modulare sui task What-Where con *random* e *blocked training*.

Quindi dei tre sistemi studiati solo l'architettura modulare è capace di mostrare una prestazione robusta in presenza di crosstalk temporale. Un limite è che solo una expert network determina l'uscita, una volta per ogni step. Quando task diversi devono essere eseguiti simultaneamente, il sistema assegna expert network diverse per imparare task diversi, perciò l'architettura modulare è vulnerabile agli effetti di crosstalk spaziale ma con una semplice modifica dell'architettura il problema è risolto.
L'architettura (Fig. 45) consiste di due expert network e due gating network: una gating network cancella i primi *m* componenti di ogni vettore di output della expert network e l'altra gating network cancella i rimanenti. Una funzione determina i valori desiderati per i primi *m* componenti del vettore di output *y* e una seconda funzione determina i valori desiderati per i rimanenti.

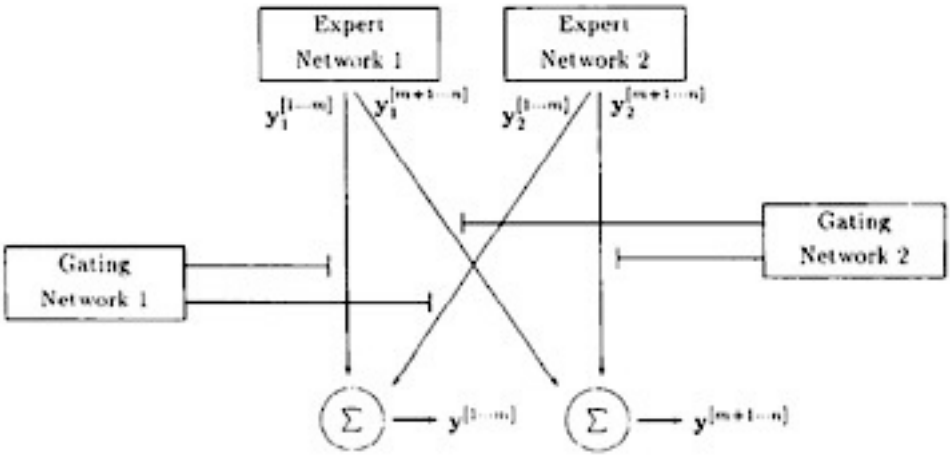


Fig. 45. Architettura modulare con più gating network ($y_1^{[1\cdots m]}$: il vettore i cui componenti sono i primi *m* del vettore di uscita della Expert Network 1 e le altre espressioni simili indicano sottovettori).

Le gating network possono assegnare la stessa expert network a entrambe le funzioni oppure diverse a ciascuna. Come nell'architettura modulare con singola gating network, la competizione determina come i pattern training sono assegnati nel corso dell'apprendimento. Per l'architettura modulare con più gating network è identico a quello di una singola gating network con la condizione che le modifiche dei pesi sono determinate indipendentemente per ciascuna gating network. Un'architettura modulare con più gating network può essere pensata come un insieme di architetture modulari separate, ognuna con una singola gating network. Condividono le expert network sebbene le loro gating network cancellano diversi insiemi di componenti di output delle expert network. Di conseguenza i componenti di output delle expert network che non sono cancellati non contribuiscono a modificare i pesi. Quindi per ogni gating network c'è un processo separato per determinare quando la prestazione dell'architettura è migliorata significativamente ed è un processo identico a quello delle architetture con gating network singole, dipende solo dall'errore sui componenti di output cancellati dalla gating network. Questo risulta nei valori diversi λ_{WTA} e λ_{NT} per ogni gating network e quindi in diverse funzioni di errore.

7.2

Scomposizione di task

Poiché la scomposizione di una funzione è un problema con vincoli, esistono più scomposizioni possibili di un task in task più semplici, tant'è che le architetture modulari con molti gradi di libertà non possono scomporre un task come desiderato (Denker e al. 1987; le Cun, 1989; Poggio & Girosi, 1989). Se ci sono ragioni di preferire una scomposizione ad un'altra, si utilizza conoscenza del dominio al fine di progettare un architettura ristretta ai tipi di funzioni che può computare, vale a dire come un architettura scomporrebbe un task nel modo desiderato.
Progettare un'architettura adatta potrebbe consentire di usare conoscenza a priori sul task che il sistema ha richiesto di eseguire, questo rappresenta un vantaggio delle architetture modulari. Per esempio, in molti task c'è una distinzione fra informazione da elaborare e informazione che imposta il contesto di elaborazione. Un altro modo di conoscenza a priori che può essere incluso nel progetto di un architettura modulare riguarda la conoscenza delle proprietà della funzione da approssimare e risulta valido quando le expert network consistono di reti con caratteristiche diverse. Quindi se un task può essere scomposto in sottotask, ciascuno ha proprietà idiosincratiche e, allora il sottotask dovrebbe essere a sua volta un sistema scomponibile in cui risorse distinte di sistema (expert) vengano assegnate a sottotask distinti. Cosi un sistema di apprendimento sarà più robusto, efficiente e, generalizzerà meglio di un sistema non modulare.
Sebbene la conoscenza del dominio sia utile nella scomposizione a priori di un task, i limiti non sono evidenti nei dati presentati al sottotask. L'assegnamento ottimale di expert a sottotask dipende dalla natura del task ma anche del sottotask. Anche se la conoscenza del dominio è usata per progettare la struttura iniziale dell'architettura modulare, è necessario per il sistema scoprire le expert ai quali assegnare esempi di apprendimento, per cui l'architettura modulare presentata fa uso di competizione per indurre una scomposizione di task. La competizione permette a reti expert di specializzarsi bene da estendere il loro dominio.

7.3

Modularità *task-dependent*

Le prestazioni dei sistemi neurali dipendono dalla loro architettura e la modularità è il risultato o il prodotto di un processo di creazione.
Il task di trovare una rete adatta per un dato problema è spesso risolto per mezzo della computazione evolutiva. Su questa base, gli algoritmi evolutivi sono spesso progettati in maniera da favorire le architetture modulari. Per quali tipi di task sono favorite le architetture modulari? Ci sono misure generali ed espressive per la modularità?
Le ipotesi già testate hanno confermato che i problemi modulari sono risolti meglio da reti modulari che non modulari.
Qui sono considerati diversi task con eredità *Lamarckian* e *Darwinian* al fine di testare la *task-dependency* dell'ipotesi. In particolare per verificare se il *fast learning* (basato su gradiente) aumenta la modularità.
Bullinaria (2001) e Di Ferdinando e al. (2001) hanno studiato l'evoluzione *non-Lamarckian* della modularità architetturale in reti con un solo layer nascosto del problema "What-Where" trovando che le architetture modulari si sviluppano per migliorare l'apprendimento, cioè le reti evolvono in modo tale che alcuni dei neuroni nascosti sono connessi solo ai neuroni di output "What" o solo "Where". Per studiare la modularità sono considerate due tipi di applicazione di reti ottimizzate, indicate come due tipi di task: una denotata *accurate model* progettata per risolvere un unico problema, per esempio rappresenta un mapping input-output indotto da un insieme di dati campione con la massima precisione possibile; l'altro task è un *fast learner*, un architettura che può imparare un dato problema in breve tempo. Questo è più sensibile quando affronta un numero di task diversi ma connessi.
Il problema da apprendere cambia in ogni generazione ma rimane separabile. Per rendere i risultati dell'apprendimento veloce confrontabili alla modellazione del task accurato, sono eseguite prove di ottimizzazione fast learner senza cambiare il problema in ogni generazione, cioè lo stesso problema deve essere appreso ripetutamente più velocemente possibile.
I simboli (≠) (=) indicano se il problema cambia periodicamente o meno. E' stato scelto un problema separabile con due diverse classi di problemi ($m = 2$) e tre input booleani ciascuno ($n = 6$). Per ogni processo evolutivo sono scelte a caso etichette di classi corrispondenti agli input di ogni problema tale che metà dei pattern di input appartengono a ciascuna classe.
L'insieme dei dati contiene tutte le possibili combinazioni di input (per es. $2^n = 64$ pattern diversi), "true" e "false" codificati come 1 e -1. Le reti neurali sono con 6 input, 2 neuroni di output lineari e il valore assoluto della misura di modularità è influenzato dal numero di neuroni nascosti e connessioni.
In particolare non tutte le reti modulari ($M \approx 1$) evolvono in caso di leggera pressione selettiva verso reti piccole. Per evitare *side-effect*, la dimensione delle reti si mantiene costante.
Le simulazioni sono eseguite con 10 neuroni nascosti con funzione di attivazione sigmoidale e 40 connessioni e le reti risultanti abbastanza grandi per risolvere il problema ma sparse per consentire l'evoluzione di moduli separati.
Inoltre non c'è un numero massimo di layer nell'architettura, la sola restrizione è che ogni neurone nascosto deve essere connesso, direttamente o indirettamente ad almeno un input e a un neurone di output. L'apprendimento è eseguito con iRprop$^+$ (Igel & Husken, 2002), versione migliorata dell'algoritmo Rprop (Riedmiller & Braun, 1993) che è un potente algoritmo di apprendimento basato su gradiente discontinuo. Lo scopo dell'apprendimento è la minimizzazione dell'errore quadratico medio $E^{(mse)}$. Il risultato di classificazione della rete è calcolato da mapping con output positivo a "true" e negativo a "false". L'errore di classificazione $E^{(class)}$ è dato dal rate di classificazioni incorrette. Ancor prima della prima generazione, 10 individui (μ) sono inizializzati a caso.

Le 40 connessioni distribuite a caso su tutta l'architettura e i pesi inizializzati con valori nell'intervallo [-0.2, 0.2]. Il solo operatore impiegato per generare la prole ($\lambda = 70$) è lo spostamento casuale di singole connessioni (per es. una connessione è cancellata e inserita successivamente ad una posizione casuale e inizializzata nuovamente a caso) cosi il numero di connessioni rimane costante.
I genitori della generazione successiva sono selezionati per mezzo della (μ, λ)-selezione, ad es. la migliore uscita μ della prole λ forma la successiva popolazione genitore.
A seconda dell'ottimizzazione del task (modello accurate/fast) gli algoritmi differiscono leggermente. Nel caso precedente in ciascuna generazione ogni individuo è addestrato per 200 cicli. Nell'evoluzione *Lamarckian* l'apprendimento inizia dai pesi che sono codificati dopo l'apprendimento nella generazione precedente.
Nell'evoluzione *Darwinian* è codificata nel genoma degli individui solo l'architettura e i pesi per l'apprendimento inizializzati una seconda volta nell'intervallo [-0.2, 0.2]. La funzione fitness include solo gli errori dopo l'apprendimento:

$$\phi^{(\text{accurate model})} = -\left(E^{(\text{class})} + \alpha_2 E^{(\text{mse})}\right)$$

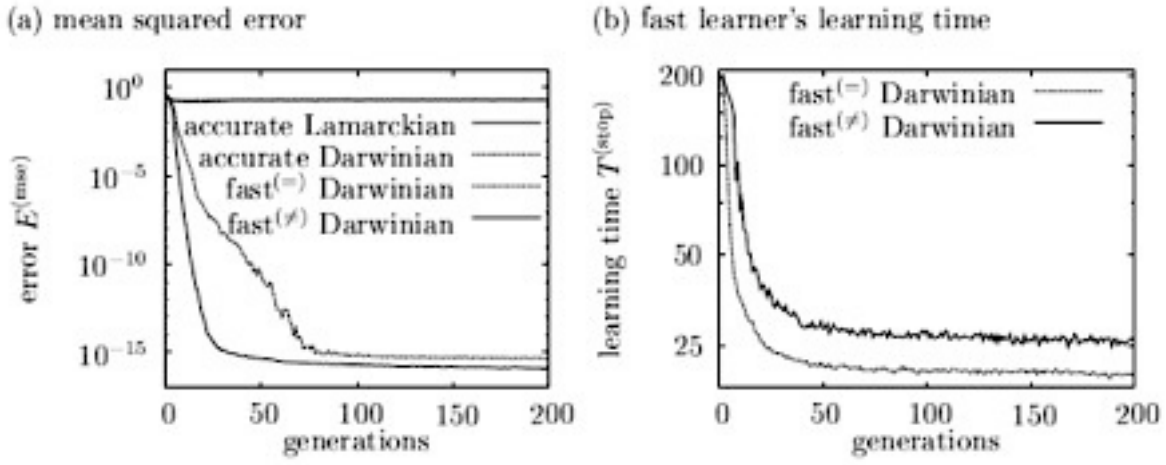


Fig. 46. Termini della funzione di fitness

Con evoluzione fast learner, i pesi di ogni individuo di ogni generazione sono inizializzati nuovamente a caso nell'intervallo [-0.2, 0.2] (stile Darwinano) e l'apprendimento avviene purchè ci siano ancora pattern erroneamente classificati a sinistra, ma non più di un numero massimo di 200 cicli ($T^{(max)}$). In questo caso, la fitness è data da:

$$\phi^{(\text{fast learner})} = -\left(E^{(\text{class})} + \alpha_1 \left(T^{(\text{stop})} / T\,(\text{max})\right) + \alpha_2 E^{(\text{mse})}\right)$$

$T^{(stop)}$: rappresenta il ciclo quando l'apprendimento reale si è fermato.
La scelta di $\alpha_1 = 10^{-4}$ e $\alpha_2 = 10^{-8}$ fa si che $E^{(class)}$ sia il più forte; $E^{(mse)}$ debole influenza la fitness.
Gli stessi esperimenti sono eseguiti anche con condizioni diverse: considerando neuroni di uscita sigmoidali anziché lineari e la funzione di costo $E^{(mse)}$ con valori 0.1 e 0.9, con funzione di costo $E^{(mse)}$ con valori 0 e 1 e con funzione di costo Cross-Entropy con valori 0 e 1.
Nel primo caso i risultati sono gli stessi, nel secondo e terzo caso, i risultati non possono essere riprodotti. Questo spiega anche i risultati di Bullinaria (2001) secondo il quale l'uso di una funzione di costo $E^{(mse)}$ con valori 0.1 e 0.9 porta a diversi gradi di modularità della funzione di costo Cross-Entropy.

Quindi si distingue fra criteri di ottimalità differenti (apprendere una classificazione massimale veloce oppure una regressione massimale accurata in un dato tempo) e diversi schemi di eredità (Lamarkiana con eredità di pesi appresi e Darwiniana senza eredità).
La modularità aumenta con l'efficienza delle reti e l'aumento è *task-dependent*: l'apprendimento veloce senza eredità di pesi rafforza lo sviluppo di architetture modulari, mentre la regressione accurata (con e senza eredità di pesi) rafforza lo sviluppo di una configurazione modulare di pesi.
Generalmente le reti modulari sono efficaci per risolvere problemi modulari e il bisogno di modularità diventa molto forte, poi l'apprendimento più veloce e robusto diventa parte della definizione di un task.

CAPITOLO 8

La modularità risolve l'interferenza [*]

Nelle reti modulari ciascun peso è sempre coinvolto in un unico task: alcune connessioni sono coinvolte nel realizzare un primo task, mentre altre nel realizzare un secondo mentre in reti non modulari alcune connessioni sono coinvolte nella realizzazione di un primo task, altre di un secondo e altre ancora entrambi. Quest'ultimo è importante in un'architettura non modulare e solleva il problema dell'interferenza. Essa deriva dal fatto che la corretta realizzazione di un primo task possa richiedere che il peso iniziale durante l'apprendimento sia aumentato mentre la corretta realizzazione del secondo possa richiedere invece che il peso venga diminuito. Tale problema è presente in reti non modulari che devono apprendere molteplici task.
Una soluzione all'interferenza è un'architettura modulare in cui ogni modulo è dedicato alla soluzione di un task. Inoltre è attiva a livello genetico in entrambe le architetture e può completamente escludere l'adattamento di un carattere a causa di collegamento genetico con mutazioni deleterie che interessano un altro carattere.
L'interferenza genetica è un meccanismo che riduce la possibilità di evoluzione di entrambi i tipi di rete. Può essere esaminata bene nel modello What-Where. Esso se modellato da una rete non modulare può sorgere interferenza con effetto sulle prestazioni e potrebbe derivare dal fatto che nel corso dell'apprendimento, il peso di una delle connessioni aumenta per realizzare un task e diminuisce per realizzarne altri, quindi la modularità è una soluzione al problema.
Una rete modulare è una rete in cui alcune connessioni sono dedicate a un task e non giocano nessun ruolo in altri, mentre altre dedicate al secondo task e non usate per il primo. Questo tipo di modularità risolve il problema dell'interferenza perché i pesi di ogni task possono essere aumentati o diminuiti nella maniera adatta senza interferire con la realizzazione degli altri.
Rueckl et al. (1989) nel loro modello What-Where hanno dimostrato che quello che distingue l'architettura modulare dalla non modulare è il pattern delle connessioni proiettanti dalle unità nascoste a quelle di output. Nell'architettura non modulare, tutte le unità nascoste proiettano ad ognuna delle unità di output, mentre nell'architettura modulare le unità nascoste proiettano solo alle unità di output What e le rimanenti solo alle unità di output Where. Più unità nascoste sono dedicate al task What che al task Where poiché il task What è più complesso.
Quindi si può concludere che nelle architetture non modulari, tutte le connessioni più basse sono coinvolte in entrambi e questo può derivare dall'interferenza, considerato che nell'architettura modulare alcune delle connessioni più basse sono responsabili solo per il task What e le rimanenti solo per il Where.
Una rete addestrata con algoritmo backpropagation impara i task meglio, se l'architettura è modulare piuttosto che non modulare. Nei lavori di Rueckl e al. (1989) l'architettura è fissa e cablata a priori. Di Ferdinando e al. (2001) hanno condotto una serie di esperimenti con architettura non definita a priori ma evolve in una popolazione di reti con algoritmo genetico. I pesi per l'architettura ereditata geneticamente sono appresi nel corso della vita di ogni singola rete con il backpropagation di 100 reti, ciascuna con un pattern di connessioni più alte generate a caso.

[*] R. Calabretta "Genetic interference reduces the evolvability of modular and non modular visual neural networks" in Philosophical Transactions of The Royal Society: Biological Sciences, 2010.

Iniziando con pesi casuali assegnati alla nascita a ciascuno dei 100 individui, ciascuno impara i pesi per entrambi i task ed è assegnato un valore di fitness (somma degli errori dei due task, con segno meno) alla fine dell'apprendimento. I 20 individui con fitness più alta (errore totale più piccolo) sono selezionati per la riproduzione e ciascuno genera 5 figli. I 20 x 5 = 100 nuovi individui costituiscono la generazione successiva e ogni prole eredita la stessa architettura del genitore ma non i pesi. Dopo un numero di generazioni, la fitness media risulta buona, confrontata con i risultati di Rueckl. L'architettura che emerge alla fine dell'evoluzione si avvicina all'architettura modulare di Rueckl, con 14 unità nascoste per il task What e 4 per il Where. L'evoluzione trova spontaneamente la stessa architettura trovata da Rueckl e al., la quale è ottimale per i task.
Si può immaginare di affidare l'evoluzione a entrambi i task per trovare la migliore architettura e pesi adatti, cioè tutto è ereditato geneticamente e non vi è nessun apprendimento nel corso della vita. In queste condizioni è impossibile creare reti in grado di risolvere i task (Di Ferdinando e al., 2001), l'errore alla fine rimane considerevole, dato che qualche insieme di pesi sarà adatto per risolvere entrambi i task ma solo rispetto a qualche architettura e non per ciascuno.
Negli esperimenti (Di Ferdinando e al., 2001) il genotipo ereditato codifica architettura e pesi e quando un individuo eredita un genotipo, implica che architettura e peso codificati siano adatti a ogni altro, altrimenti il genitore non dovrebbe esser selezionato per la riproduzione. Se una mutazione cambia l'architettura ereditata, l'insieme dei pesi codificati nello stesso genotipo potrebbe non essere più adatto per quell'architettura. Questo può spiegare perché l'algoritmo genetico non è in grado di trovare una soluzione per entrambi i task se è ereditato tutto geneticamente e l'evoluzione deve identificare architettura e pesi.
Nei pochi casi in cui l'evoluzione è in grado di trovare l'architettura adatta, poi non è in grado di trovare anche i pesi. Nelle poche repliche della simulazione in cui l'architettura evoluta si avvicina a quella ottimale (14 unità nascoste per il modulo What e 4 per il Where) l'errore finale non si avvicina a zero (Di Ferdinando e al., 2001) e il problema va oltre il fatto che le mutazioni possano spezzare un co-adattamento di evoluzione di architettura e pesi.
L'evoluzione può essere non in grado di trovare pesi adatti per le architetture modulari, quindi si potrebbe affidare l'evoluzione al task di trovare i pesi di un architettura fissata e ottimale e i genotipi ereditati possono codificare solo i pesi, mentre l'architettura è fissa (è l'architettura ottimale di 14 unità nascoste per il task What e 4 per il Where), non è codificata nel genotipo e non evolve. L'evoluzione risulta in grado di trovare i pesi adatti per un'architettura fissa come alternativa al backpropagation.
Con l'algoritmo genetico per evolvere i pesi di architetture fisse, si deve risolvere solo uno dei task e l'architettura tende a essere non modulare.
L'evoluzione sarà in grado di trovare i pesi per un'architettura modulare fissa controllando il fatto che si devono risolvere entrambi i task?
Varie simulazioni con architettura fissa e modulare, ottimale quella dei task What-Where e con l'algoritmo genetico che necessita solo di evolvere i pesi, hanno prodotto i risultati (Fig. 47): per entrambe le architetture (modulare e non), l'algoritmo non è in grado di trovare pesi per i due task.

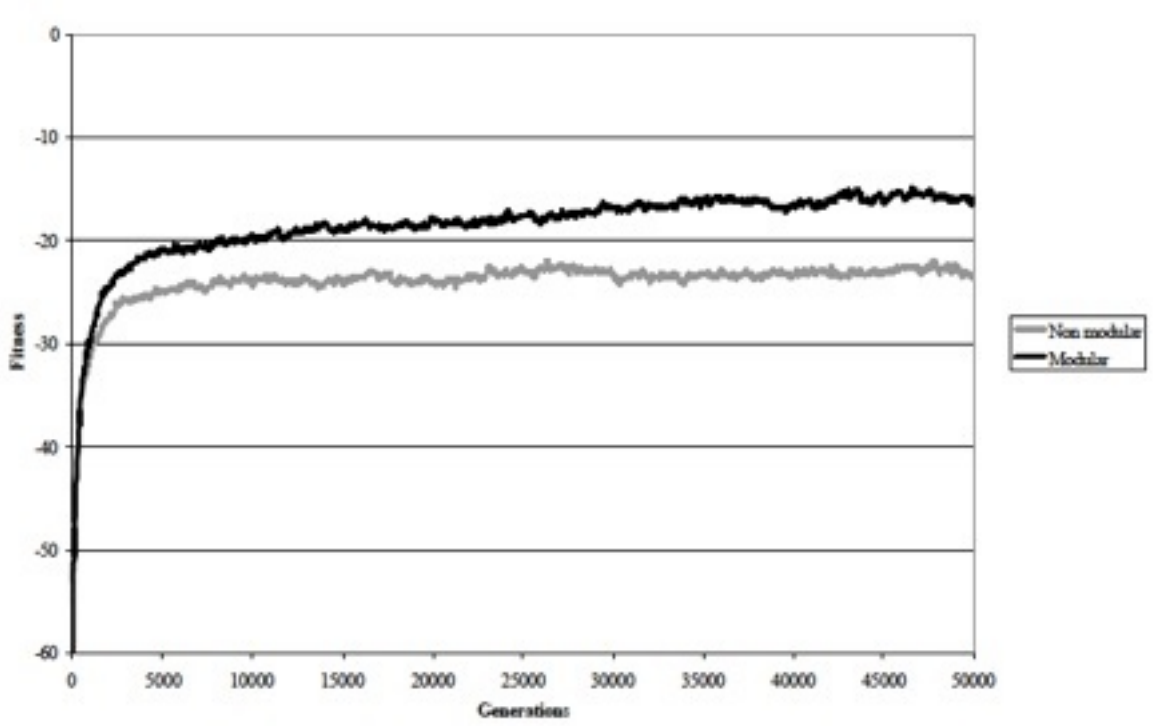


Fig. 47. Fitness media dei task in 50.000 generazioni per architetture modulari e non, con architettura fissa ed evolvono solo i pesi.

E' importante notare che mentre l'interferenza neurale è presente nell'architettura non modulare perché tutte le connessioni servono entrambi i task, non risulta vero per l'architettura modulare. Nell'architettura modulare sono usate connessioni diverse per i due task e possono essere modificate separatamente dal processo evolutivo senza alcuna interferenza. Quindi l'interferenza non spiega i risultati non buoni ottenuti con architettura modulare ma possono essere spiegati con un altro tipo di interferenza non a livello neurale ma genetico.
L'interferenza genetica avviene perché diverse parti del genotipo codificando moduli diversi, sono collegate insieme nello stesso genoma ed ereditate insieme.
Perché il collegamento genetico può avere conseguenze negative sull'evoluzione?
Quando i pesi di un'architettura modulare sono modificati con il backpropagation, esso sarà in grado di trovare i pesi per entrambi i task e l'errore totale approssimato a zero (Rueckl e al., 1989). Questo perché con il backpropagation è come utilizzare due algoritmi di apprendimento separati, uno per il task Where e l'altro per il What. Per ogni ciclo, calcola due errori separati, uno per ogni modulo e modifica i pesi di ciascuno separatamente sulle basi di un errore. L'errore totale sarà la somma dei due calcolati con l'algoritmo genetico e i pesi evoluti piuttosto che appresi.
C'è differenza sostanziale fra le due procedure: con backpropagation ciascun modulo è informato separatamente della prestazione di rete su ogni task; con il genetico, la fitness che determina quali individui riproducono e quali no, è una valutazione globale di ogni individuo sommando insieme l'errore dei due task. Questo crea le condizioni per l'interferenza genetica causata dal collegamento genetico.
Cosa può accadere quando un genotipo ripartito in due parti, una codificante i pesi per il modulo What e l'altro i pesi per il Where, è riprodotto e soggetto a mutazioni genetiche casuali?
Si immagina che una mutazione cambi la parte What del genotipo risultante in una prestazione migliorata del modulo What. Se questa è la sola mutazione genetica che interessa il genotipo, la prestazione globale risulterà migliorata, l'individuo si riprodurrà e la mutazione favorevole sarà conservata nella popolazione. Tuttavia, si considera il caso in cui una mutazione favorevole avvenga sulla parte What del genotipo e contemporaneamente una sfavorevole avvenga sulla parte Where. Il problema che il processo di selezione non può "vedere" la mutazione vantaggiosa. Se l'effetto della fitness globale di entrambe le mutazioni è deleterio, la mutazione vantaggiosa sarà

eliminata da selezione naturale. Perciò se rate e mutazioni deleterie sono alti, si potrebbe impedire l'adattamento di un carattere e in questo caso il task What (Fig. 48).

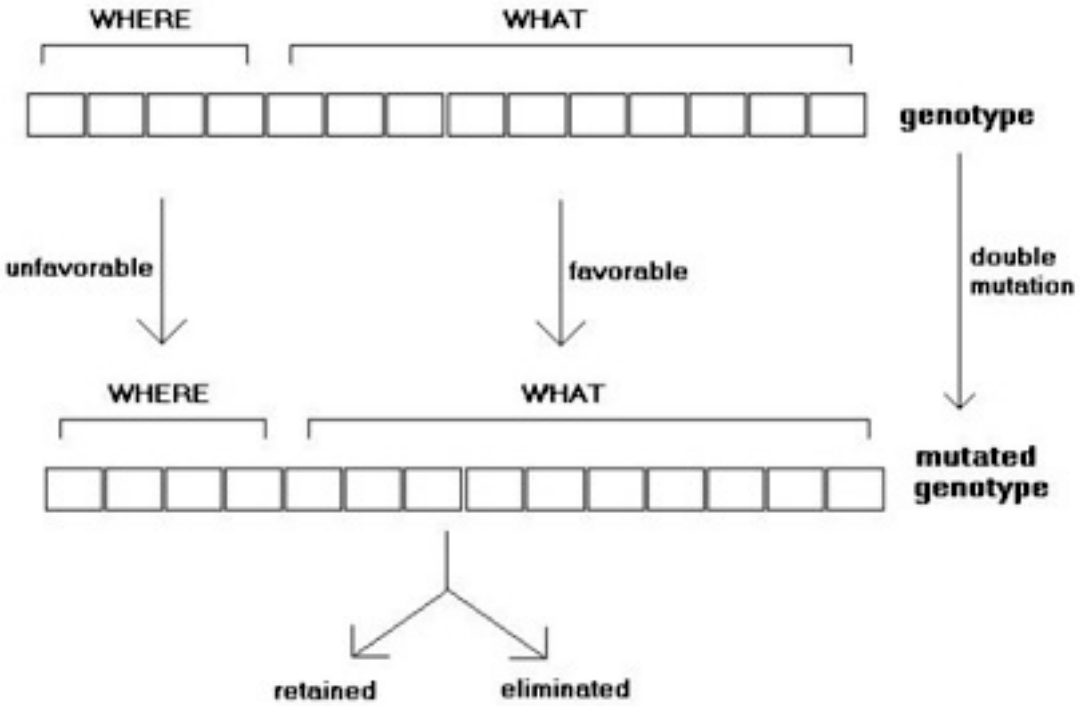


Fig. 48. Interferenza genetica. Mutazioni in conflitto (una favorevole e una sfavorevole) avvengono su parti separate del genotipo che codifica moduli distinti.

Si potrebbe provare a eliminare il collegamento genetico e valutare le prestazioni dei due task separatamente. L'algoritmo genetico evolve due popolazioni separate di individui, una in cui ogni individuo deve risolvere solo il task What e l'altro con individui che devono risolvere solo il task Where. Con il backpropagation non c'è interferenza fra i due task (in architetture modulari) perché i pesi del modulo What sono modificati solo sulle basi dell'errore sul task What e similmente per il Where. Queste condizioni sono ricreate con l'algoritmo genetico con due popolazioni distinte.
Il genotipo di una popolazione codifica i pesi solo per il modulo What e gli individui si riproducono solo sulle basi della loro prestazione sul task What. Lo stesso accade per il Where. Perciò ogni popolazione è valutata solo per l'interferenza genetica di un task, nello stesso genotipo è impossibile. In queste simulazioni l'algoritmo genetico è applicato separatamente a due distinte popolazioni: una con architettura adatta per il task What (25 unità input, 14 nascoste, 9 di output) e la fitness riflette sulla prestazione dell'individuo solo su quel task, mentre l'altra popolazione ha un'architettura adatta per il task Where (25 unità input, 4 nascoste, 9 di output) e la fitness cattura solo una prestazione dell'individuo su questo task. In queste circostanze può non esserci interferenza genetica e l'algoritmo genetico evolve i pesi adatti per entrambi i task. In entrambe le popolazioni, l'errore finale sul rispettivo task è vicino a zero.
Oppure si potrebbe provare a manipolare il rate di mutazione. Un rate più alto dovrebbe produrre con grande probabilità che una mutazione favorevole in un modulo sia realizzata da una mutazione sfavorevole nell'altro. Aumentando il rate da 0.0016% a 10% per popolazioni in cui un individuo deve eseguire entrambi i task e per popolazioni separate in cui un individuo esegue solo uno dei due si hanno i risultati (Fig. 49): con entrambe le condizioni c'è un aumento di fitness quando i rate aumentano da un valore di 0.0016% a 0.3%, con valore 0.3% entrambi sono risolti in entrambe le condizioni, oltre l'equilibrio di fitness diminuisce. Mentre la fitness è la stessa nelle due condizioni con rate molto bassi. La diminuzione oltre la soglia dello 0.3% è molto più grande per popolazioni in cui i moduli sono entrambi codificati nello stesso genotipo. Quando due moduli distinti sono nello stesso genotipo, le mutazioni di un modulo possono essere più adatte per un task e nello stesso

tempo possono produrre un modulo adatto ma più piccolo per l'altro. La probabilità che ciò avvenga, aumenta con l'aumentare dei rate.

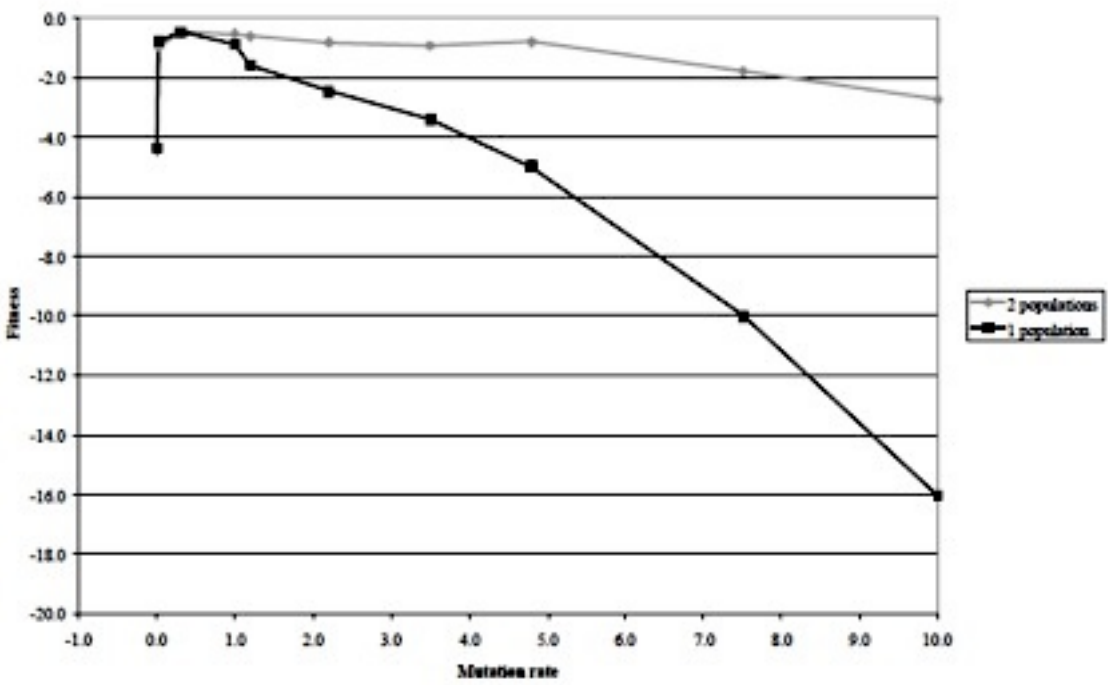


Fig. 49. Fitness media alla fine dell'evoluzione di popolazioni con rate aumentati con genotipo che codifica entrambi i task (popolazione 1) e uno dei due (popolazione 2). La fitness delle "popolazioni 2" è la somma della fitness della popolazione What e Where.

8.1

Interferenza genetica

Secondo Rueckl e al. (1989) un individuo che deve risolvere entrambi i task, li deve risolvere simultaneamente in risposta allo stesso input anche se differiscono in complessità o difficoltà. Il task What è più complesso o difficile del task Where. La difficoltà che differenzia i due task si manifesta nel fatto che il task Where è acquisito (evoluto o appreso) prima e raggiunge una prestazione quasi perfetta dell'altro.
Con il backpropagation le reti esibiscono quello che è chiamato "effetto età dell'acquisizione": la prestazione sui task che sono stati appresi prima, più di altri, è migliore di quella sui task acquisiti più tardi (Ellis & Lambon Ralph, 2000; Smith e al.). Questo effetto è osservato anche nelle simulazioni di popolazioni di reti che evolvono per risolvere i task What-Where. Il task Where è acquisito prima del What, nel senso che il relativo errore diminuisce più rapidamente (nelle generazioni prima) che del What. Solo quando l'errore del Where raggiunge un livello basso, quello del What inizia a diminuire significativamente e tale distribuzione risulta più bassa sull'errore finale del Where che nel What.
Nelle prime generazioni, gli individui sono selezionati per la riproduzione a vari stage evolutivi in termini del loro errore sul task Where mentre l'errore sul What è ignorato. Solo quando nelle generazioni successive, l'errore del Where risulta basso, le differenze fra individui in questo errore diventano trascurabili e l'errore del task What diventa il principio di base di quegli individui selezionati per la riproduzione, però è troppo tardi per acquisire bene il task What e alla fine dell'evoluzione rimane più alto dell'errore sul Where.
Si potrebbe considerare interferenza genetica ristretta ai casi in cui un task è acquisito prima di un altro, poichè uno potrebbe essere più facile dell'altro. Sono stati condotte due ulteriori insiemi di

simulazioni. Un modo per ristabilire una parità fra entrambi i task è assegnare valori più adatti al What che rimane più difficile del Where, ma si decide che sia più importante in termini di cambiamenti riproduttivi. Tale importanza si può compensare per le sue grandi difficoltà ed entrambi possono esser acquisiti insieme e bene. In questa situazione la tendenza dell'evoluzione è di concentrarsi nelle prime fasi evolutive sul task Where, più facile. Per cambiare il valore di fitness dei due task viene modificata la formula usata per selezionare gli individui per la riproduzione: l'errore nel task Where è aggiunto all'errore del What per produrre l'errore totale. Questo errore totale con segno meno è la fitness di un individuo che decide se l'individuo è uno dei 20 individui che riproducono.
In un nuovo insieme di simulazioni viene cambiata la formula per dare più peso all'errore sul What confrontato con il peso dell'errore sul Where e la nuova formula di fitness diventa:

Fitness Totale = 5 x Fitness What + Fitness Where

Cosi il task What anche se più difficile diventa 5 volte più importante nel decidere quali individui sono selezionati per la riproduzione. Perciò, l'evoluzione si concentra sul task What finchè sono raggiunti livelli alti di prestazione, ma quando successivamente l'evoluzione passa al Where è troppo tardi. Questo impedisce di evolvere verso il raggiungimento di buoni livelli di prestazione nel task Where.
Una soluzione migliore al problema può essere ottenuta se il task più difficile What fornisce pesi più adatti del task più facile Wheree si adotta la seguente formula di fitness:

Fitness Totale = 3.5 x Fitness What + Fitness Where

Questo mostra che ci sarà un aumento parallelo della fitness per entrambi i task dall'inizio dell'evoluzione (il valore 3.5 riflette il rapporto fra 14 unità nascoste What e 4 unità nascoste Where). Anche se i task sono acquisiti simultaneamente, l'interferenza genetica impedisce il raggiungimento di un errore finale soddisfacente su entrambi i task (Fig. 50).

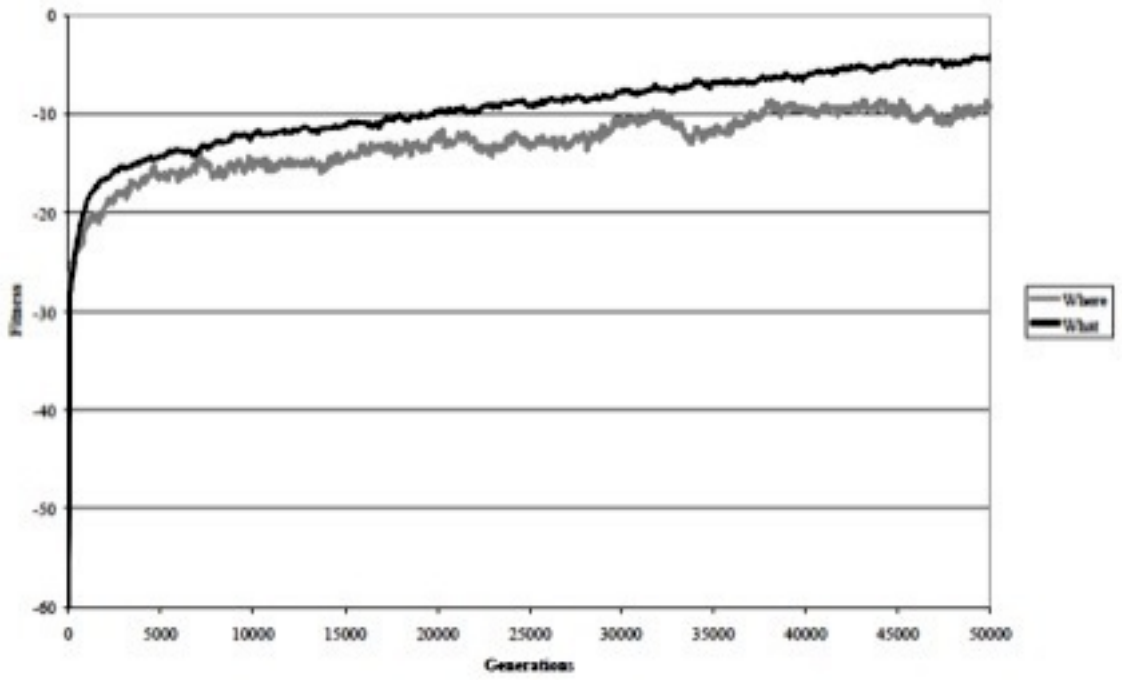


Fig. 50. Fitness media di 50.000 generazioni separate per i task. Entrambi evolvono insieme per la modifica della formula di fitness. Il valore più grande del task What compensa la grande difficoltà rispetto al Where.

L'interferenza genetica è un fenomeno generale non ristretto a casi di task diversi. Sono state condotte altre simulazioni in cui si devono risolvere due task identici. In una popolazione, si deve risolvere il task Where due volte in risposta allo stesso input e in un'altra popolazione si deve risolvere due volte il task What. In due volte/popolazione l'architettura Where include due moduli, ciascuno con 4 unità nascoste e 2 insiemi separati di unità di output.
Con il backpropagation, la rete fornisce lo stesso input di apprendimento Where due volte per i due moduli. Lo stesso per le due volte/popolazione What con due moduli identici What con 14 unità nascoste ciascuno. La fitness è la somma dei due errori Where o dei due What.
Nelle nuove simulazioni potrebbe emergere interferenza genetica, poichè i genotipi ereditati codificano due insiemi separati di pesi, uno per ciascuno dei due moduli identici Where o What. Successivamente i due task che devono essere evoluti in ogni popolazione sono di difficoltà identica.
La natura diversa dei task non può spiegare gli effetti dell'interferenza genetica. I risultati mostrano che anche con task di difficoltà identica che evolvono insieme, l'interferenza genetica ostacola l'evoluzione (Fig. 51, 52).

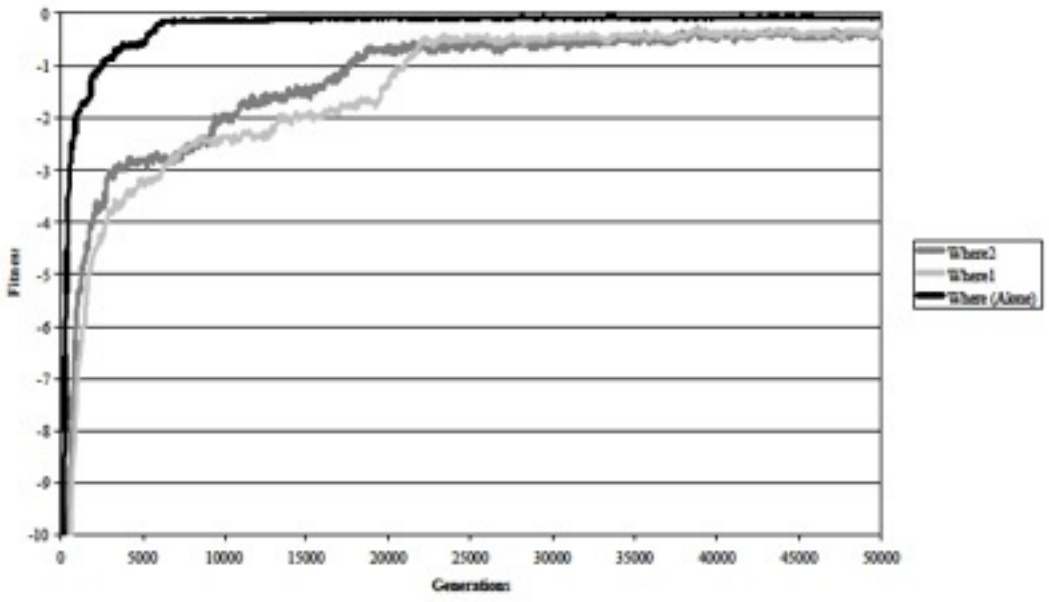


Fig. 51. Fitness media di 50.000 generazioni in una popolazione che deve risolvere i due task Where confrontata con una popolazione che ne risolve uno solo.

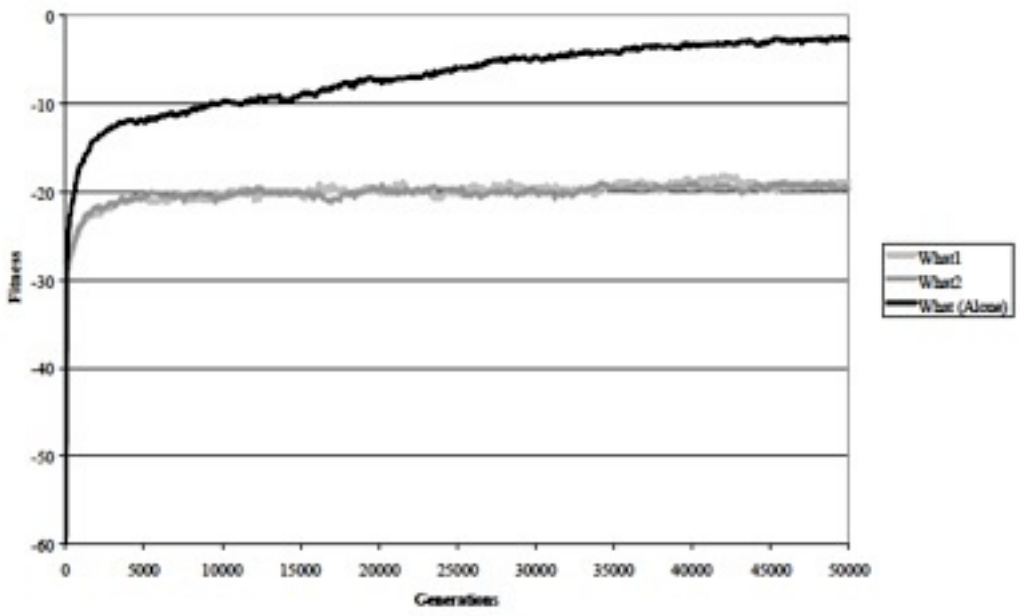


Fig. 52. Fitness media di 50.000 generazioni in una popolazione che deve risolvere 2 task What confrontata con una popolazione che ne risolve uno solo.

Tali risultati spiegano il comportamento di organismi semplici che tende a essere geneticamente ereditato mentre quelli più complessi esibiscono comportamenti appresi nel corso della vita.
La semplicità/complessità può essere definita in termini di numero di task diversi. Un organismo è semplice se il suo pattern adatto richiede l'esecuzione di un numero limitato di task diversi. Un organismo è complesso se per sopravvivere e riprodursi deve esser in grado di eseguire molti task diversi. Mentre organismi semplici tendono ad avere sistemi nervosi non modulari, la necessità di evitare interferenza richiede che il sistema nervoso di un organismo complesso sia modulare.
L'esistenza di moduli separati per task distinti consente ai pesi di non interferire con altri.
Perché molti sistemi modulari tendono a contare sull'apprendimento in vita come meccanismo per trovare i pesi adatti dei vari moduli, invece di affidare all'evoluzione?
Perché i pesi di molti sistemi modulari di organismi complessi sono appresi piuttosto che ereditati, considerato che negli organismi più semplici possono essere ereditati?
Elman e al. (1996) hanno proposto che l'architettura dei sistemi nervosi è prevalentemente ereditata geneticamente. Di Ferdinando e al. (2001) hanno dimostrato che la soluzione migliore per reti che devono eseguire entrambi i task è avere l'evoluzione che si occupa di trovare l'architettura modulare adatta e l'apprendimento trova i pesi per l'architettura ereditata.
La ricombinazione sessuale è un meccanismo che può ridurre l'interferenza genetica combinando insieme parti di genomi di due individui distinti, i quali possono esser stati entrambi migliorati da mutazioni. Sembra sia in grado di risolvere il problema del collegamento genetico solo per architetture con pochi moduli separati.

8.2

Interferenza nelle prime fasi dell'apprendimento

L'interferenza compare prevalentemente nelle prime fasi dell'apprendimento. Con apprendimento backpropagation, il peso di qualche connessione da cambiare è proporzionale all'errore dell'unità che arriva la connessione.
Questo errore *E* è la differenza fra unità osservata e valore di attivazione desiderato:

$$E = t_i - a_i$$

t_i : valore di attivazione desiderato; a_i : valore di attivazione osservato.
In generale, le connessioni cambiano il loro valore corrente quando gli errori della rete sono grandi. Quando sono piccoli, sono richieste meno connessioni. Una conseguenza è che nelle reti non modulari, l'interferenza è maggiore quando gli errori sono grandi e sono richieste connessioni per cambiare in maniera sostanziale il valore dei loro pesi. Più una connessione deve cambiare il proprio valore, più critica è l'interferenza se i cambiamenti richiesti vanno in direzioni opposte.
Quando gli errori di una rete neurale sono grandi? La risposta è nelle prime fasi dell'apprendimento.
L'interferenza in reti non modulari che devono imparare due task contemporaneamente sarà più elevata nelle prime fasi dell'apprendimento. L'errore quadratico medio SSE, è la somma degli errori quadratici di tutte le unità di output per tutti i pattern di input:

$$SSE = \tfrac{1}{2} \Sigma E_i^2$$

Nella curva SSE (Fig. 53) per reti non modulari che imparano entrambi i task, l'errore totale diminuisce rapidamente nelle prime fasi dell'apprendimento e più gradualmente nelle fasi successive ed è ciò che si osserva in reti non modulari che apprendono i task What-Where delle simulazioni di Rueckl e al.

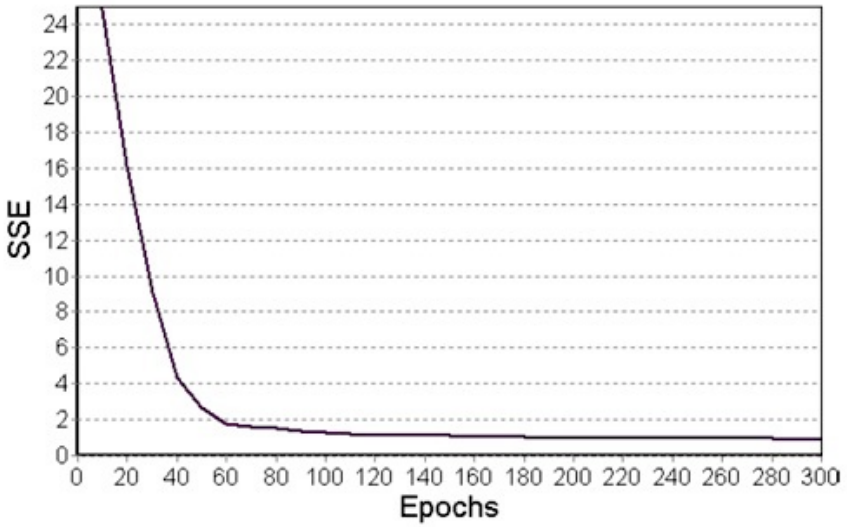


Fig. 53. Curva SSE per reti non modulari. Media di 10 repliche.

Nelle curve delle repliche singole (Fig. 54) si osserva che in alcune repliche la rete riesce con successo nell'apprendimento del task (SSE si avvicina a zero) ma nella maggior parte dei casi non è cosi. Ancora una volta non si comprende perché ciò accade.

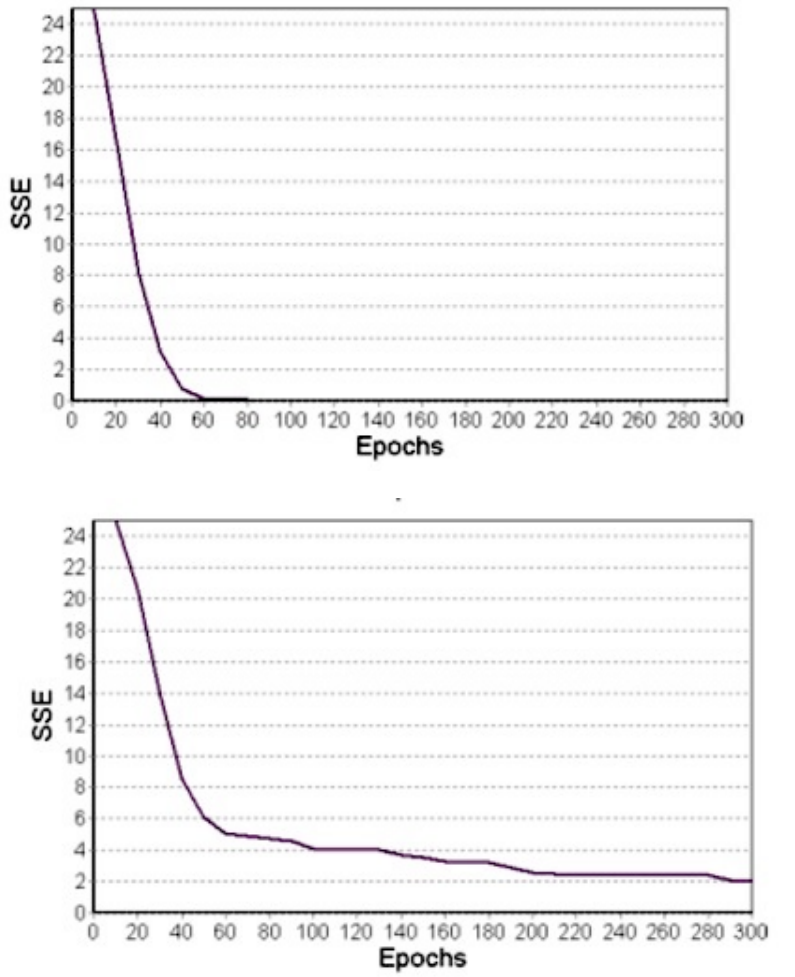


Fig. 54. Curva SSE per 2 repliche per reti non modulari. Nella prima SSE si avvicina a zero, nella seconda rimane ancora grande dopo 300 epoche.

Se invece di interrompere l'apprendimento dopo 300 epoche, si continua fino a 10.000, SSE è vicino a zero sempre (Fig. 55).

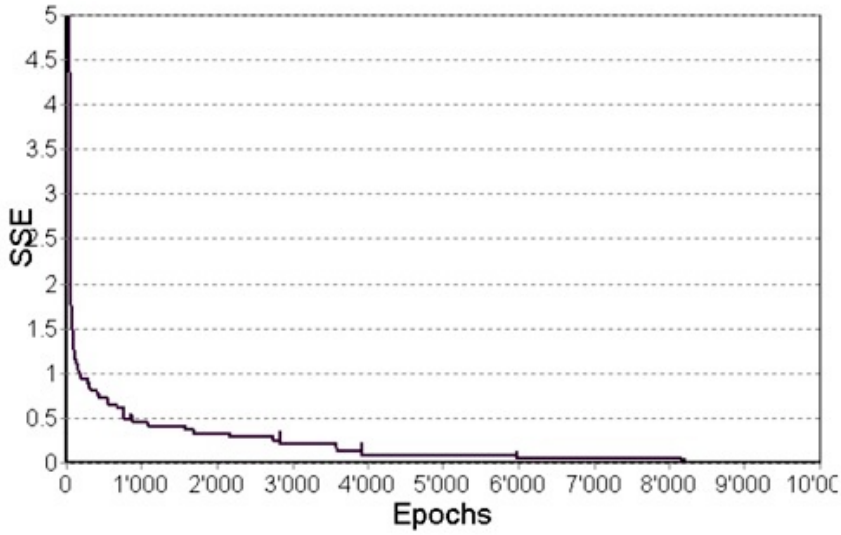


Fig. 55. Curva SSE per reti non modulari per 10.000 epoche (media di 10 repliche).

8.3

Apprendere due task uno dopo l'altro

In tutte le simulazioni descritte, le reti neurali apprendono i due task insieme. Ora immaginiamo che una rete neurale apprende i due task non insieme contemporaneamente ma uno dopo l'altro.
La rete comincia ad imparare un task. Dopo che è stato appreso il primo task, ad es. SSE per il task vicino a zero, è introdotto il secondo task e la rete inizia a impararlo. Da notare che quando inizia l'apprendimento del secondo, l'apprendimento del primo non viene interrotto.
Questa forma di apprendimento sequenziale è applicata a reti modulari con risultati non diversi dal caso dove sono appresi contemporaneamente.
E' stato già osservato che le reti modulari sono in realtà costituite da due distinte sottoreti (moduli) che non condividono ciascun peso singolo. Se i task sono appresi insieme o in maniera sequenziale è irrilevante, l'SSE per entrambi è vicino a zero in entrambi i casi.
La questione interessante è: l'apprendimento sequenziale consente a reti non modulari di acquisire i task altrettanto bene?
La risposta è si e riduce anche se non completamente l'interferenza. La Fig. 56 mostra come questo accade.

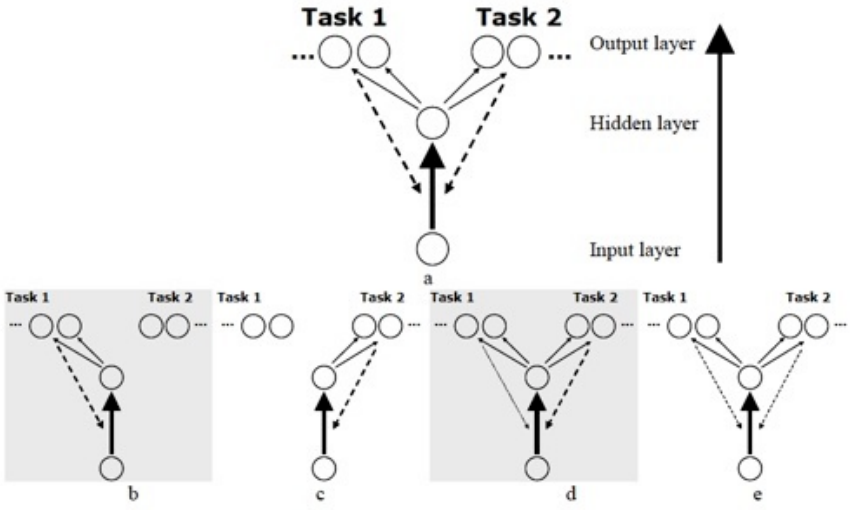


Fig. 56. Interferenza fra due task in architettura non modulare. Linee in grassetto: pesi. Linee tratteggiate: messaggi di errore (linea tratteggiata marcata, messaggio di errore più grande). **(a)** Fase iniziale di apprendimento simultaneo dei due task. **(b), (c), (d), (e)** 4 Fasi di apprendimento sequenziale.

Quando la rete impara due task contemporaneamente, è nelle prime fasi dell'apprendimento che l'interferenza raggiunge la sua massima forza perché i messaggi di errore sono più grandi (Fig. 56a). Al contrario, nelle prime fasi di apprendimento sequenziale c'è un solo task da apprendere e senza interferenza (Fig. 56b).
Quando il primo task è appreso (SSE vicino a zero), il secondo è aggiunto ma non vi è ancora alcuna interferenza fra i due poichè il primo ha già imparato e non invia alcun messaggio di errore (Fig. 56c).
Quando l'algoritmo di apprendimento inizia a modificare i pesi per imparare il secondo task, le prestazioni nel primo possono diminuire. Poiché i pesi più bassi sono condivisi dai due task, i cambiamenti di pesi derivanti dall'apprendimento dell'input per il secondo task possono influenzare la prestazione della rete sul primo e viceversa. Non appena la prestazione del primo task

diminuisce, esso riparte per inviare messaggi di errore e saranno più piccoli rispetto a quelli del secondo (Fig. 56d).
Come l'apprendimento procede, la prestazione nel primo task può continuare a diminuire per un po' ma nel frattempo la prestazione nel secondo aumenta, per cui i messaggi di errore dai due task non potranno mai essere entrambi molto grandi contemporaneamente e non ci sarà mai una forte interferenza che compare quando i due task sono appresi contemporaneamente (Fig. 56e).
Riassumendo, l'apprendimento sequenziale evita la presenza contemporanea di due messaggi di errore grandi insieme e riduce, anche se non elimina l'interferenza. Una conseguenza è che dato che il primo task è appreso senza interferenza mentre il secondo con qualche interferenza seppur ridotta, è meglio imparare prima il task più difficile del più facile.

CAPITOLO 9

Ambienti favorevoli all'evoluzione della modularità [*]

Un approccio diverso per favorire l'evolvere della modularità è considerare ambienti dinamici simili agli ambienti statici mostrando perché l'informazione strutturale di una rete neurale modulare possa essere utilizzata direttamente limitando l'effetto di altri parametri.
Ci sono due tipi diversi di ambienti: nel primo un individuo non aspetta di imparare un dato task ma si adatta a un task *Related* che condivide alcune delle sue caratteristiche con il task originale; nell'altro scenario un individuo aspetta di imparare molti più task complessi in maniera incrementale.
Mentre adattandosi al *Related Task* l'individuo aspetta di imparare una funzione della forma *g(f1, f2)* e poi adattarsi a *g'(f1, f2)*, mentre adattandosi a un task complesso per un task complesso si aspetta di imparare una funzione *g(f1, f2, f3)* dopo l'apprendimento di *g(f1, f2)*.
Si assume una rete neurale modulare con topologia coordinata, dopo l'apprendimento è richiesta la prima funzione in fase 1 per adattarsi alla seconda funzione in fase 2 (Fig. 57).
Con *Incrementally Complex Task*, agli individui è consentito crescere. Di seguito è illustrato come sistemi modulari e non, si adattano a tali ambienti considerando funzioni booleane. In fase 1, i moduli *f1* e *f2* si specializzano nelle corrispondenti sotto funzioni.

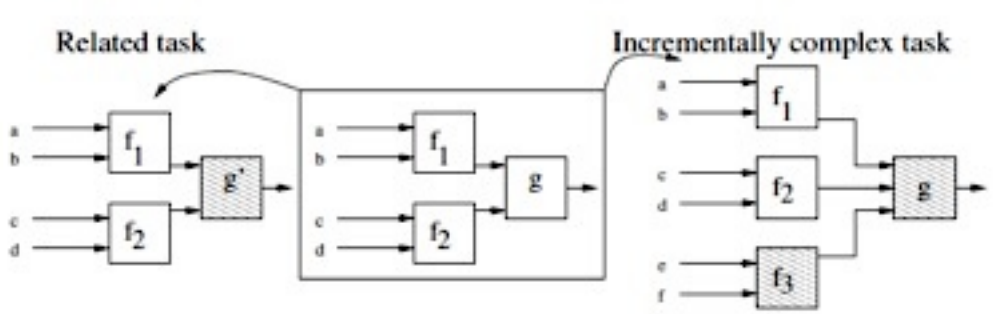


Fig. 57. Rete modulare si adatta da *g(f1, f2)* (centro) al task *related g'(f1, f2)* (sinistra) o a un task più complesso *g'(f1, f2, f3)* (destra). I moduli ombreggiati sono etichettati "new".

Per utilizzare l'informazione strutturale della rete modulare, bisogna usare questi due moduli in fase 2. Imparando la nuova funzione, i moduli perdono la loro specializzazione. Cambiando la funzione, gli errori corrispondenti derivati sono inizialmente grandi con grandi risultati nei cambiamenti dei parametri in tutti i moduli di rete, perciò si elimina la specializzazione.

[*] N. Kashtan, E. Noor, U. Alon "Varying environments can speed up evolution" in Proceedings of the National Academy of Sciences, 2007
V.R. Khare, B. Sendhoff, X. Yao "Environments Conducive to Evolution of Modularity" in Parallel Problem Solving fron Nature – PPSN IX, Lecture Notes in Computer Science, N.4193, 2006.
R.A. Jacobs, M.I. Jordan "Computational Consequences of a Bias toward Short Connections" in Journal of Cognitive Neuroscience MIT, N.4, 1992.

Una delle soluzioni a questo problema è fissare i parametri dei due moduli e tenerli fissi durante l'apprendimento in fase 2. Un approccio meno restrittivo consiste nel non fissare i parametri ma provare a controllare i cambiamenti all'inizio della fase 2.
Ostacolando grandi cambiamenti in questi parametri, ci sarà la combinazione di un modulo con un cambiamento in modo tale da adattarsi a moduli già specializzati. Anche in assenza di informazione sulla relazione fra le due funzioni, è preferibile il secondo approccio perché la rete può imparare la seconda funzione che non sarebbe possibile usando il primo. Per implementare tale approccio, c'è bisogno di un parametro associato a ognuno dei pesi, controllando l'importanza dei loro cambiamenti.
Con l'algoritmo *Improved Resilient Back-propagation* (IRPROP) già c'è un tale parametro chiamato *step-size*. Durante l'apprendimento il suo valore assoluto decresce nel tempo e nelle fasi successive si hanno valori molto piccoli. All'inizio della fase 2 dell'apprendimento, i moduli che si vogliono ostacolare sono etichettati "*re-used*" e altri come "*new*". Il paramero step-size è usato alla fine della fase 1 per essere inizializzato nella fase successiva dell'apprendimento ha valori piccoli nella rete. All'inizio della fase 2 dell'apprendimento i moduli sono etichettati come "re-used" e altri come moduli "new". Poi si usano step-size alla fine della fase 1 per inizializzarli per la successiva fase di apprendimento per tutti i parametri nei moduli "re-used", mentre i moduli "new"hanno i loro step-size inizializzati normalmente.
Questo è un modo di trattare l'apprendimento in sistemi modulari, per mezzo di cui vecchi moduli mantengono le loro caratteristiche, mentre i nuovi iniziano con caratteristiche nuove. Con tale schema di inizializzazione, gli step-size corrispondenti ai due moduli etichettati differentemente esibiscono comportamenti tipici (Fig. 58). In fase 2, gli step-size associati a modulo "new" iniziano da un valore costante alto, mentre gli step-size associati a modulo "re-used" iniziano con valori molto bassi, prima aumenta, poi si va verso la diminuzione e questo indica alcune modifiche dei moduli "re-used".

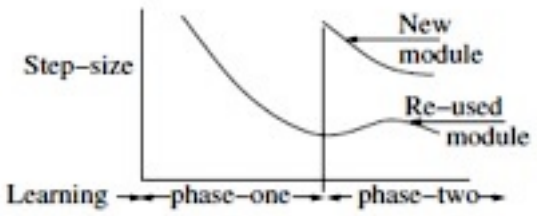


Fig. 58. Cambiamenti dei valori assoluti di step-size associati con parametri di vari moduli nel corso dell'adattamento al task related.

9.1

Adattabilità vs *Related Task*

Per capire il ruolo della modularità in un ambiente dinamico si considera un esempio: il problema XOR-OR.

Dopo l'apprendimento della funzione XOR *Composite* $((a \oplus b) \oplus (c \oplus d))$ in fase 1, una rete deve adattarsi in fase 2 alla funzione OR *Composite* $((a \oplus b) + (c \oplus d))$; a, b, c e d sono variabili booleane; + e $\oplus$ sono le funzioni OR e XOR, queste condividono sottofunzioni comuni e la combinazione delle funzioni OR *(g)* e XOR *(g')* sono differenti.

Per questo problema, si confronta una struttura modulare e una completamente connessa con l'algoritmo IRPROP-modificato utilizzando anche entropia trasversale e funzioni di errore quadratico medio.

La struttura completamente connessa è una rete RBF (Rete di funzioni di base radiali) e la rete modulare è una combinazione di tre reti RBF più piccole, ognuna rappresentante un modulo.

Il numero di unità nascoste sono scelte anche per avere uno stesso numero di parametri liberi in entrambe le reti.

L'apprendimento della fase 2 inizia con i pesi appresi dopo la fase 1 ed entrambe consistono di 100 epoche. Il risultato è che la struttura modulare si adatta molto meglio di quella completamente connessa, se si usa un algoritmo di apprendimento a discesa più ripida.

Con IRPROP entrambe le strutture si adattano bene. Con IRPROP modificato, la struttura modulare è migliore.

9.2

Adattabilità vs *Incrementally Complex Task*

Per confrontare l'adattabilità fra una struttura completamente connessa e una modulare per task che diventano sempre più complessi si considera il seguente esempio costituito di tre fasi: nella prima fase il task deve imparare *f1*; nella seconda deve imparare *g(f1, f2)* e nella terza deve imparare *g(f1, f2, f3)*. $f1 = a \oplus b$, $f2 = c \oplus d$, $f3 = e \oplus f$ e la funzione combinazione g è OR. Ancora a, b, c, d, e, f sono variabili booleane e $\oplus$ è lo XOR.

Nella prima fase non c'è modularità perciò si inizia con due reti RBF completamente connesse. Nella seconda fase, una di queste è cresciuta in maniera modulare attraverso i nuovi input che vanno in un modulo separato e l'altra è cresciuta aggiungendo più unità nascoste nella rete. Entrambe ancora una volta sono cresciute in modo simile nella terza fase. In ogni fase, il numero di parametri totale nelle due strutture è mantenuto uguale.

Nelle curve di apprendimento (Fig. 59) per le due strutture con funzione di scarto quadratico medio NRMSE si osserva che nella prima e seconda fase è presente un errore di apprendimento mentre nella terza fase un errore test. La struttura modulare usa l'informazione strutturale al proprio interno ed esegue meglio di una struttura completamente connessa. Tale differenza di prestazioni aumenta con l'aumentare dei sottotask all'interno del task completo ed è osservata senza considerare la funzione errore usata per l'apprendimento.

Quindi il vantaggio della modularità nelle reti neurali è molto più visibile in ambienti dinamici. Con semplici task dinamici si possono mostrare benefici dell'adattabilità da modularità e anche la parziale spiegazione dell'abbondanza di modularità in sistemi complessi naturali, per i quali ambienti dinamici possono essere parzialmente responsabili.

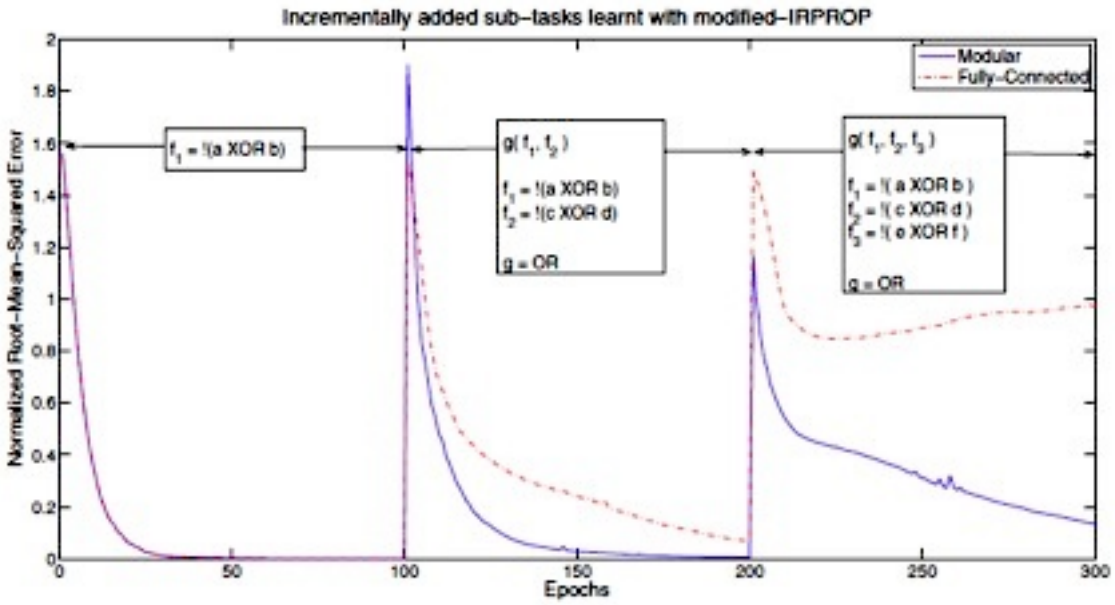


Fig. 59. NRMSE medio di 30 simulazioni a epoche diverse con apprendimento IRPROP-modificato per la struttura modulare e con apprendimento IRPROP per la struttura completamente connessa.

9.3

Ambienti variabili accelerano l'evoluzione

Goal variabili nel tempo hanno impatto sulla velocità di evoluzione che rappresenta il numero di generazioni necessarie ad una popolazione iniziale casuale di raggiungere un dato goal. Rispetto a goal fissi possono accelerare l'evoluzione. L'aumento maggiore si verifica in *modularly varying goal* MVG diversi con obiettivi che cambiano nel corso del tempo, ma ciascun nuovo obiettivo condivide alcuni dei sottoproblemi con il precedente. La velocità aumenta con la complessità del goal. Gli MVG spingono le popolazioni fuori dai massimi locali di fitness e le guidano verso soluzioni modulari capaci di evolvere.
Ambienti variabili nel tempo possono influenzare proprietà dei sistemi evoluti quali struttura, robustezza, possibilità di evoluzione e mapping genotipo-fenotipo. In particolare gli MVG in cui ciascuno dei nuovi goal condivide alcuni sottoproblemi con il goal precedente generano spontaneamente sistemi con struttura modulare.
Un aumento di velocità è osservato con MVG e in alcune condizioni con *random varying goal* RVG.

9.3.1

Casi di studio

Confrontiamo l'evoluzione con goal fisso e con quattro scenari di goal che variano nel tempo: MVG e tre scenari diversi RVG. In MVG, sono stati considerati goal che possono essere scomposti in ulteriori sottogoal. Il goal cambia di volta in volta in modo che ciascuno dei nuovi goal condivide alcuni dei sottogoal con il precedente. Per esempio, i due sottogoal descritti dalle funzioni *f(x,y)* e *h(w,z)* possono essere combinati con una terza funzione *g* per formare goal modulari con quattro input: *G = g(f(x,y), h(w,z))*. Sono generate variazioni modulari di tale goal modificando una delle funzioni. Per esempio, i sottogoal composti dalla funzione XOR: *f = x XOR y* e *h = w XOR z*. Uno dei goal può essere dato dalla combinazione di questi, usando un OR, *G1 = g(f,h) = f OR h = (x XOR y) OR (w XOR z)*. Una variazione modulare del goal *G2 = g'(f,h) = f AND h = (x XOR y) AND (w XOR z)* è generata cambiando la funzione *g = OR* a *g' = AND*. Una variazione modulare diversa è *G3 = g(f',h) = (x EQ y) OR (w XOR z)* ottenuta cambiando la funzione *f = XOR* a *f' = EQ* (*equals*). In questo modo possono essere generati tanti goal modulari.
Oltre a MVG, gli altri scenari riguardano cambiamenti periodici fra un dato goal G1 e un goal casuale R e poi indietro a G1 e cosi via. Abbiamo confrontato due scenari diversi di RVG: nel primo, i goal cambiano periodicamente fra G1 e lo stesso goal casuale R (scenario RVG_c) dove *c* indica un goal costante casuale; nell'altro è scelto un goal casuale nuovo ogni qualvolta il goal cambia (scenario RVG_v) dove *v* indica un goal che varia casualmente.
Infine, abbiamo esaminato uno scenario in cui il goal passa periodicamente da un determinato goal G1 a una situazione con nessuna selezione di fitness ma soltanto evoluzione (scenario VG_0).
Per confrontare le velocità di evoluzione con goal fissi e variabili, abbiamo usato algoritmi genetici standard, iniziando con una popolazione di reti casuali, ciascuna rappresentata da un genoma che rappresenta nodi e connessioni. L'evoluzione procede verso un goal definito: calcolare un output specifico sulla base degli input. Ogni rete calcola la fitness, ossia la frazione di tutti i possibili valori di input per cui la rete fornisce l'output desiderato. Le reti con fitness più alta forniscono più alta probabilità di replicarsi. Per modificare i genomi, sono applicate operazioni genetiche standard: mutazioni e crossover. Cosi come le generazioni avanzano, la fitness aumenta fino a raggiungere una soluzione perfetta in cui è trovato il goal. Abbiamo usato quattro tipi di modelli di rete (Fig. 68): circuiti logici booleani, reti neurali *integrate-and-fire* e reti di funzioni continue. Oltre a quelli computazionali anche un modello strutturale di RNA con algoritmi genetici per evolvere le molecole di RNA verso una struttura specifica secondaria.

Model	Building blocks	Fold-Speedup for the hardest goals (S_{max}), mean ± SE			
		MVG	RVG_v	VG_0	RVG_c
1 Logic circuits	NAND gates	**95 ± 45**	**45 ± 20**	2.5 ± 2	<1
2 Feed-forward logic circuits	NAND, OR, AND	**265 ± 150**	**160 ± 80**	**190 ± 90**	1.3 ± 0.3
3 Feed-forward neural networks	Integrate-and-fire neurons	**700 ± 450**	**10 ± 5**	1.5 ± 1	<1
4 Feed-forward circuits	Continuous functions	**60 ± 10**	3 ± 1	3 ± 2	<1
5 RNA secondary structure	Nucleotides A, U, G, and C	**25 ± 5**	<1	<1	<1

Fig. 68. Evoluzione in ambienti variabili. S_{max} : aumento di velocità dei goal più difficili (tutti i goal con $T_{FG} > G_{Max}/2$). S_{max} (media +- SE) sotto 4 scenari con goal variabili per ciascuno dei modelli. In grassetto: media (S_{max}) > 3.

Iniziamo con i circuiti logici combinatori di gate NAND sono universali che evolvono verso un goal fisso G1 costituito di 3 XOR e 3 AND su 6 input:

G1 = [(*x* XOR *y*) AND (*w* XOR *z*)] AND [(*w* XOR *z*) AND (*p* XOR *q*)].

Iniziando dai circuiti casuali, il tempo medio di evoluzione per raggiungere il goal è: $T_{FG} = 8 \times 10^4$ +- 2×10^4 generazioni (FG goal fisso). Poi è applicato lo scenario MVG con goal cambiati ogni 20 generazioni, dove ciascun nuovo goal è simile a G1 solo che una delle 3 AND è sostituita con un OR o viceversa. L'evoluzione con MVG fornisce reti capaci di adattarsi a soluzioni perfette per ogni nuovo goal in poche generazioni. Il tempo medio per trovare soluzioni perfette per G1 dalla popolazione iniziale casuale è: $T_{MVG} = 8 \times 10^3$ +- 1.5×10^3 generazioni ed è molto più piccolo del caso con goal fisso con velocità di evoluzione di un fattore circa 10 ($S = T_{FG}/T_{MVG}$).
Nonostante ogni 20 generazioni i goal cambiano, una soluzione perfetta per il goal G1 si trova più velocemente del caso con goal applicato di continuo.
Poi è stato ripetuto per goal modulari diversi in combinazioni di funzioni XOR, EQ, AND e OR. Ogni goal ha un tempo medio diverso per raggiungere una soluzione con evoluzione con goal fisso, nell'intervallo ($T_{FG} = 2 \times 10^2$ a 3×10^6). La difficoltà dei goal (tempo necessario per risolverli "da zero") è notevole e più è difficile il goal fisso, maggiore è la velocità da MVG (Fig. 69a).
Una velocità circa 100 è osservata per i goal più difficili e aumenta come legge di potenza con la complessità del goal, $S \sim (T_{FG})$ con esponente α (0.7 +-0.1). Poi sono esaminati gli altri tre scenari di goal che variano, su ciascuno dei diversi goal cambiati ancora ogni 20 generazioni.
I goal casuali sono funzioni booleane casuali. Anche con RVG_v c'è una velocità significativa, con VG_0 e RVG_c c'è un rallentamento per la maggior parte dei goal.

Il secondo modello studiato è quello di circuiti combinatori feed-forward, con tre tipi di gate logici: AND, OR, NAND. Abbiamo evoluto i circuiti verso i goal del modello 1 e abbiamo usato scenari simili a MVG trovando che MVG accelera l'evoluzione di un fattore di circa 10^3.

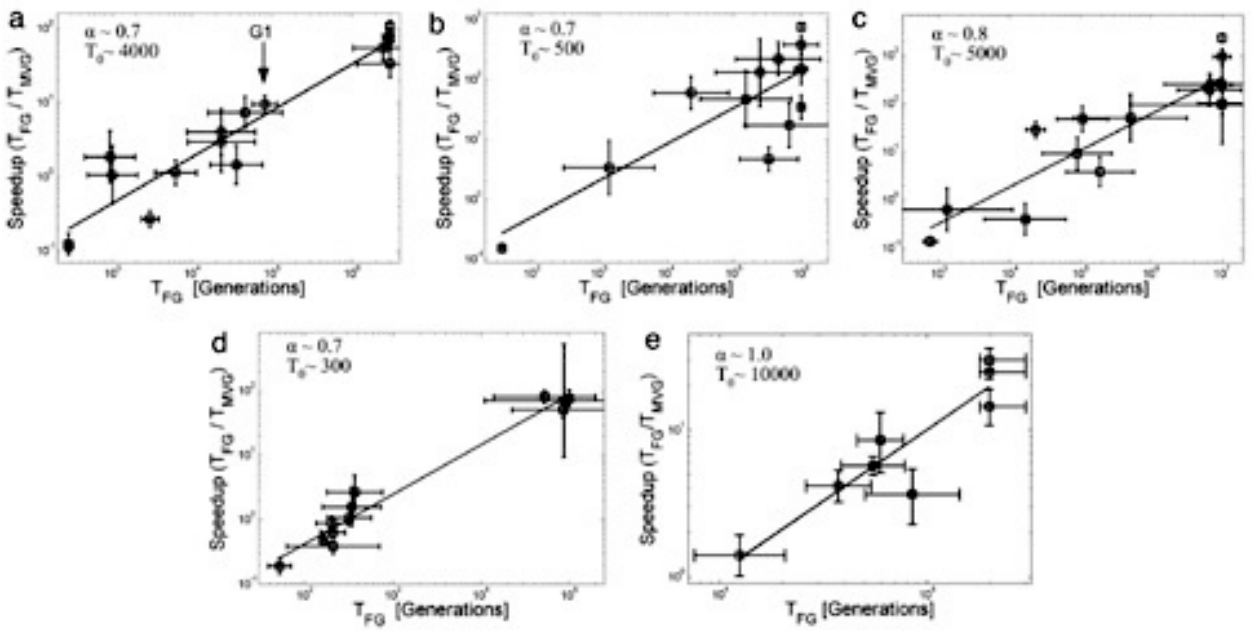


Fig. 69. Evoluzione velocità con goal variabili. 5 grafici mostrano l'aumento di velocità di modelli di sistemi diversi. Ogni grafico descrive l'aumento di velocità di evoluzione con MVG confrontato con goal fisso, verso il tempo medio di evolvere con un goal fisso. Ciascun punto rappresenta l'aumento di velocità, $S = T_{FG}/T_{MVG}$ per un dato goal. **(a)** Modello 1: circuiti logici. **(b)** Modello 2: circuiti logici feed-forward. **(c)** Modello 3: reti neurali feed-forward. **(d)** Modello 4: circuiti funzioni continue. **(e)** Modello 5: struttura secondaria RNA. L'aumento di velocità diminuisce approssimativamente come legge di potenza con esponenti α in [0.7, 1.0]. T_0 : valore T_{FG} minimo per produrre $S > 1$ (basato sulla regressione). SE_S calcolati con metodo *bootstrap*.

Più difficile è il goal e più veloce è l'aumento di velocità. Essa diminuisce come S (T_{FG}) con esponente α [0.7 +- 0.1] (Fig. 69b). RVG_v e VG_0 mostrano accelerazione più piccola che con MVG; RVG_c mostra un rallentamento per tutti i goal.
Nei due modelli di rete descritti sopra, i nodi calcolano funzioni logiche. Sono evolute anche reti con nodi che calcolano funzioni continue reali non booleane (Modelli 3 e 4) con conclusioni simili.
Il modello 3 usa reti di nodi *integrate-and-fire* con archi pesati e ogni nodo somma gli input moltiplicati per i pesi e gli output a uno dei (*fires*) se la somma supera la soglia. I goal identificano i pattern di input.
Abbiamo trovato che MVG mostra l'aumento di velocità più alto e diminuisce con la difficoltà del goal con esponente α [0.8 +- 0.1]. Nel modello 4 ogni nodo calcola una funzione continua di 2 input x e y: xy, $1 - xy$ e $x + y - xy$ e i goal sono polinomi di 6 variabili continue in [0, 1]. MVG mostra velocità più alta mentre gli altri nessuna e tale aumento dipende dalla complessità del goal ($\alpha = 0.7$ +- 0.1) (Fig. 69d).
Nel modello dell'RNA, i genomi sono sequenze di nucleotidi e il goal è una data struttura secondaria e sono usati algoritmi standard per determinare la struttura di ciascuna sequenza. La fitness di ogni molecola è definita come $1 - d/B$, con d che è la distanza strutturale del goal e B è la lunghezza della sequenza.
Con MVG, la fitness aumenta molto più velocemente che con goal fisso mentre tutti gli altri scenari mostrano rallentamento e ciò dipende dalla complessità del goal ($\alpha = 1.0$ +- 0.2) (Fig. 69e).
Sono poi esaminati gli effetti dei parametri di simulazione sull'aumento della velocità. Con MVG, si verifica per un gran numero di tempi di commutazione (numero di generazioni fra cambi di goal).

Per avere una velocità efficiente, il tempo di commutazione dei goal potrebbe essere più grande del tempo minimo necessario a ricollegare le reti per raggiungere ogni nuovo goal e, più breve del tempo necessario per risolvere un goal fisso.
Negli esempi, il primo è solitamente dell'ordine di qualche generazione, il secondo è di 10^3 generazioni o più grandi. L'intervallo di cambiamenti di tempo con notevole aumento di velocità, si estende per diversi ordini di grandezza (Fig. 70). La velocità compare per un gran numero di dimensioni di popolazione, rate di mutazione, ricombinazione e strategie di selezione.

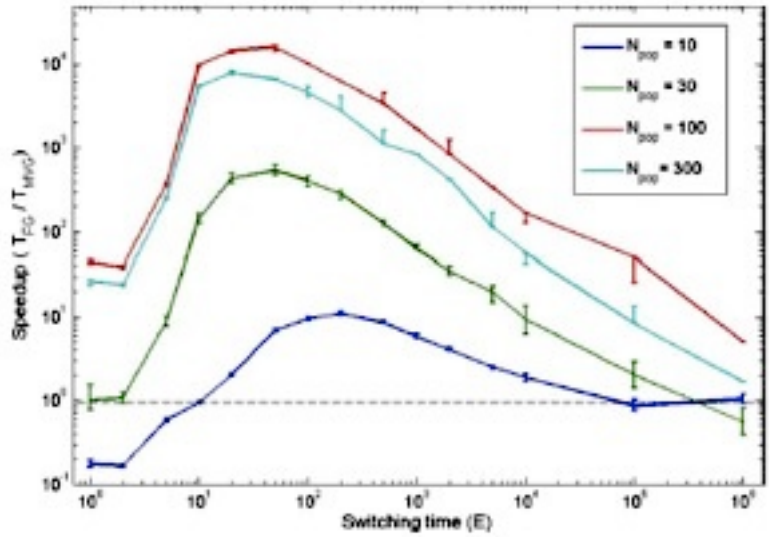


Fig. 70. Effetto della frequenza dei goal che variano sull'aumento di velocità. La velocità (+-SE) è mostrata sotto diverse frequenze di goal commutati e con varie dimensioni di popolazione (N_{pop}). Risultati per il goal G1 = (*x XOR y*) OR (*w XOR z*) con versione ridotta del modello 2 e goal a 4 input e 1 output. Con MVG il goal è commutato fra G1 e G2 = (*x XOR y*) AND (*w XOR z*) ogni *E* generazioni. Linea tratteggiata: aumento di velocità ($S = 1$).

C'è un aumento di velocità simile anche con un algoritmo in salita (*hill-climbing*) e si spiega con il fatto che è una caratteristica del paesaggio di fitness che varia, piuttosto che del preciso algoritmo usato. Empiricamente quasi tutti i goal modulari richiedono molte generazioni per risolvere l'accelerazione mostrata con MVG. Quindi l'aumento osservato con MVG causa al goal modularità, variazione di goal o entrambi?
A tale scopo, si considera un'ottimizzazione multi obiettivo in cui diverse varianti degli MVG sono rappresentati come obiettivi simultanei (non variabili) e si confronta la velocità di evoluzione di MVG con entrambi gli obiettivi: ottimizzazione ponderata multi obiettivo e ottimizzazione multi obiettivo di Pareto. Il risultato è che gli scenari multi obiettivo non mostrano aumento di velocità mentre l'equivalente scenario MVG lo mostra (Fig. 71).

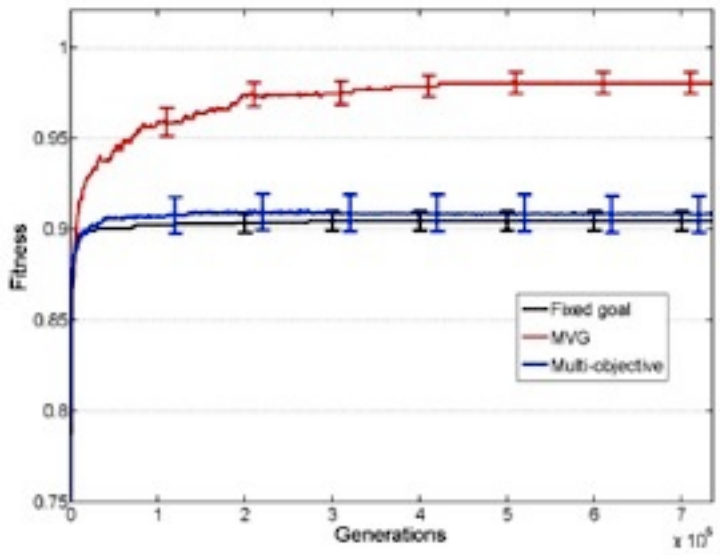


Fig. 71. Fitness in funzione del tempo in scenari MVG, goal fisso e multi obiettivo. Fitness massima nella popolazione (media +- SE) in funzione di generazioni per una versione del modello 1 con 4 input per il goal G1 = (*x XOR y*) AND (*w XOR z*). Per il caso MVG e multi obiettivo, il secondo goal è G2 = (*x XOR y*) OR (*w XOR z*). Per MVG, i dati sono per epoche e il goal G1. Per l'evoluzione multi obiettivo in cui 2 output della rete sono valutati per G1 e G2, è mostrata fitness dell'output G1. In ogni caso dati da 40 simulazioni.

Più goal modulari da soli e con nessuna variazione temporale non sono sufficienti per il verificarsi dell'accelerazione. Per risolvere questo problema è stato mappato il paesaggio di fitness di una versione del modello 2 con quattro input e un output valutando tutti i possibili genomi. Ciò consente di tracciare l'evoluzione delle reti e la distanza dalla soluzione più vicina. Durante l'evoluzione verso goal fisso la popolazione si blocca a *plateau* di fitness per tempi lunghi e sono distanti di molte mutazioni dalla soluzione più vicina. Il gradiente di fitness massima è la variazione massima di fitness su una mutazione, è zero nel plateau. Il gradiente tipicamente, punta lontano da soluzioni più vicine (Fig. 72a, b) perché c'è bisogno di un tempo lungo per trovare soluzioni in un goal fisso. RVG_v porta la popolazione in una direzione casuale, uscendo da plateau di fitness o massimi locali. MVG ha un vantaggio: ogni volta che il goal cambia si genera un gradiente positivo locale per il nuovo goal (Fig. 72c) che va nella direzione di una soluzione per il nuovo goal. Cosi la popolazione raggiunge rapidamente un'area con soluzioni per i due goal nelle immediate vicinanze (Fig. 72c, d). In tale area quando il goal cambia, le reti trovano rapidamente una soluzione per il nuovo goal già dopo poche mutazioni (Fig. 72d).

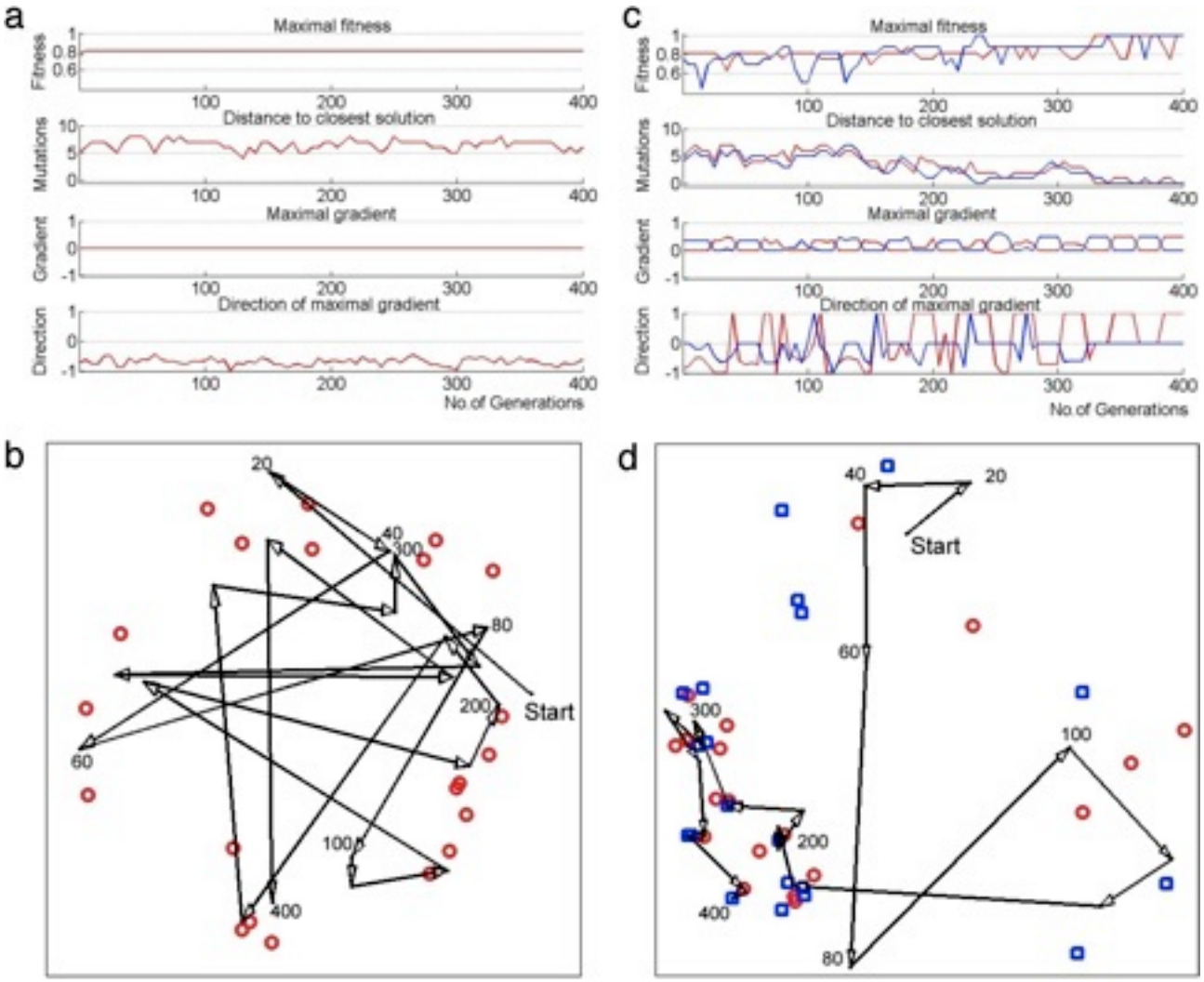


Fig. 72. Traiettorie e paesaggio di fitness con evoluzione MVG e goal fisso. **(a)** Evoluzione di una versione ridotta del modello 2, verso il goal fisso G1 = (x XOR y) OR (w XOR z). Sono mostrate le proprietà della rete migliore ad ogni generazione: fitness; distanza dalla soluzione più vicina (numero richiesto di mutazioni per raggiungere la soluzione più vicina); massimo gradiente di fitness; direzione media del gradiente massimo. -1 dalla soluzione più vicina; +1 verso la soluzione più vicina. **(b)** Traiettoria rete migliore ogni 20 generazioni (frecce). Traiettoria mappata su due dimensioni mediante *scaling* multidimensionale (tecnica che organizza i punti in uno spazio di dimensione bassa pur mantenendo le loro distanze migliori originali in uno spazio ad alta dimensione) Cerchi rossi: soluzioni più vicine. Numeri: generazioni. **(c)** Lo stesso come in (a) ma evoluzione verso MVG, cambiando fra G1 e G2 = (x XOR y) AND (w XOR z) ogni 20 generazioni. Le proprietà del circuito migliore con goal G1 e G2 sono in rosso e blu. **(d)** Traiettoria di evoluzione con MVG. Cerchi rossi: soluzioni G1 più vicine. Quadrati blu: soluzioni G2 più vicine.

Tali risultati della velocità di evoluzione sono illustrati schematicamente (Fig. 73): con goal fisso la popolazione spende la maggior parte del tempo di diffusione sul plateau o bloccato a massimi locali (Fig. 73a). Con MVG, massimi locali o plateau in uno dei goal corrispondono ad aree con gradiente positivo di fitness per il secondo goal (Fig. 73b). Su molti goal che cambiano, si forma una "rampa" sul *landscape* combinato costituito di due landscape di fitness che spingono la popolazione per i due goal verso un area con picchi vicini (Fig. 73b) e sembra che tale effetto sia diverso da una forza puramente casuale e potrebbe collegarsi alla struttura modulare delle soluzioni con MVG con moduli.

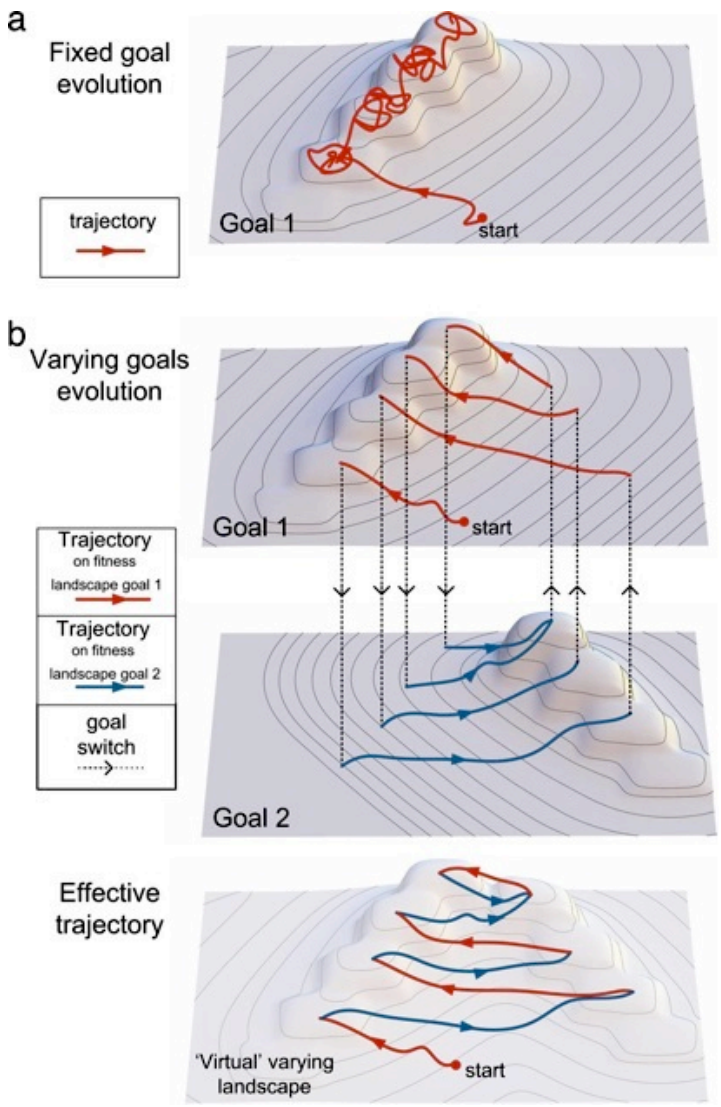


Fig. 73. L*andscape* di fitness dell'evoluzione con goal fisso e MVG. **(a)** Traiettoria dell'evoluzione con goal fisso dove la popolazione tende a trascorrere lunghi periodi in massimi locali o *plateau*. **(b)** Traiettoria con MVG. Frecce tratteggiate: goal cambiati. Un gradiente continuo positivo efficace su landscape di fitness alternati porta a un area con massimi globali in prossimità di entrambi i goal.

Quindi goal che variano possono accelerare l'evoluzione, in particolare MVG sembra velocizzare l'evoluzione in tutti i modelli e con aumento massimo. La velocità aumenta fortemente con la complessità del goal e non tutti i tipi di goal che variano mostrano tale aumento. RVG_v mostra aumento ma non sempre, dipende dal modello e dal paesaggio di fitness. Al contrario, RVG_c e VG_0 non mostrano alcun aumento in quasi tutti i casi.
I risultati evidenziano la capacità di accelerare l'evoluzione di MVG sulla base di variazioni fra goal che condividono sottogoal, piuttosto che fra goal che non lo fanno.
L'evoluzione naturale si verifica solitamente in ambienti che cambiano nel tempo e nello spazio e sono spesso modulari.

9.3.2

Metodi

Per evolvere i quattro modelli di rete e il modello di RNA sono usati algoritmi genetici standard. La popolazione di individui N_{pop} inizializzata da genomi binari casuali di lunghezza di B bit (sequenze di nucleotidi di lunghezza B per l'RNA). In ogni generazione, gli L individui con fitness più alta passano invariati alla generazione successiva. Quelli non buoni sostituititi da una nuova copia degli individui buoni. Coppie di genomi di individui non buoni sono ricombinati con probabilità di crossover P_c e quindi ogni genoma mutato a caso con probabilità di mutazione P_m e anche in assenza di ricombinazione ($P_c = 0$) si ottengono conclusioni valide. Ciascuna simulazione eseguita fino a fitness pari a 1 raggiunta per il goal o per tutti i goal nel caso di MVG. Se non raggiunta in G_{max} generazioni, T è fissato a G_{max}.

9.3.3

Modelli

I circuiti logici combinatori del modello 1 sono composti di 26 *gate* NAND a 2 input. Sono consentiti feed-back e i goal a 6 input e funzioni booleane a 1 o 2 input composte da funzioni XOR, EQ, AND e OR. Il goal è G = F(M1,M2,M3) dove M1, M2 e M3 funzioni XOR a 2 input o EQ.
F fatto di funzioni AND e OR. Il goal variabile cambia in maniera probabilistica ogni 20 generazioni applicando modifiche a F.
Il genoma binario (Fig. 74) di 256 bit è costituito di 27 geni e una singola interfaccia di output (270 bit e 28 geni, in circuiti a 2 input). Ogni gene è costituito di due campi di input che codificano la connessione di ingresso.

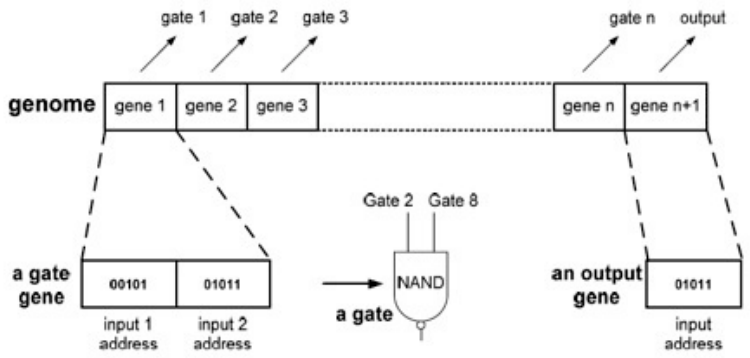


Fig. 74. Genoma del Modello 1. Genoma binario di n + 1 gate: n geni che codificano i gate (n = 26) e un singolo gene che codifica l'output.

Calcolo della fitness: goal funzioni booleane. Ogni goal definito da una tabella di verità con 6 input ($2^6 = 64$). Ogni circuito valutato su tutte le possibili combinazioni di valori in input e la fitness è il numero di voci corrette nella tabella di verità. Un circuito perfetto ha fitness pari a 1 e una penalità di fitness (0.05) è data per ogni gate aggiuntivo su un numero predefinito di gate reali (con un percorso diretto verso l'output).
Gli scenari *varying goal* simulati sono tre:

- il goal cambia fra 8 goal con sei input e un output G1-G8 della forma:
  G = F (M1,M2,M3)
  M1 = x XOR y

M2 = *w* XOR *z*
M3 = *p* XOR *q*

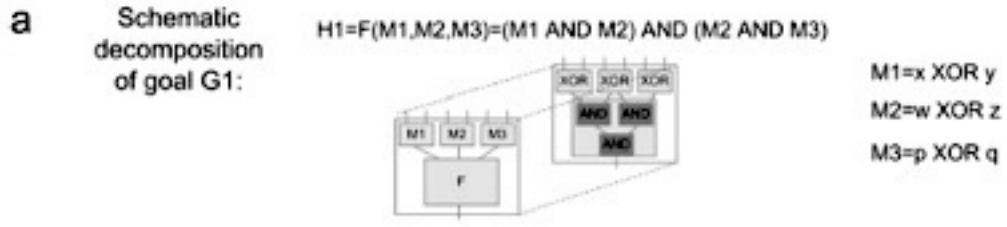
G1 = (M1 AND M2) AND (M2 AND M3)
G2 = (M1 AND M2) AND (M2 OR M3)
G3 = (M1 OR M2) AND (M2 AND M3)
G4 = (M1 OR M2) AND (M2 OR M3)
G5 = (M1 AND M2) OR (M2 AND M3)
G6 = (M1 AND M2) OR (M2 OR M3)
G7 = (M1 OR M2) OR (M2 AND M3)
G8 = (M1 OR M2) OR (M2 OR M3)

Ogni cambiamento di goal ha imposto un cambiamento di un singolo "modulo" e la funzione F è costituita di tre "moduli" (combinazioni di AND e OR).

- Il goal cambia fra 4 goal con sei input e due output G9-G12 della forma:
  G = {F1 (M1,M2,M3); F2 (M1,M2,M3)}
  M1, M2 e M3 definiti come nel caso precedente

G9 = {M1 AND M2; M2 AND M3}
G10 = {M1 AND M2; M2 OR M3}
G11 = {M1 OR M2; M2 AND M3}
G12 = {M1 OR M2; M2 OR M3}

- I goal G13-G16 definiti come nel caso precedente ma con:
  M1 = *x* EQ *y*
  M2 = *w* EQ *z*
  M3 = *p* EQ *q*



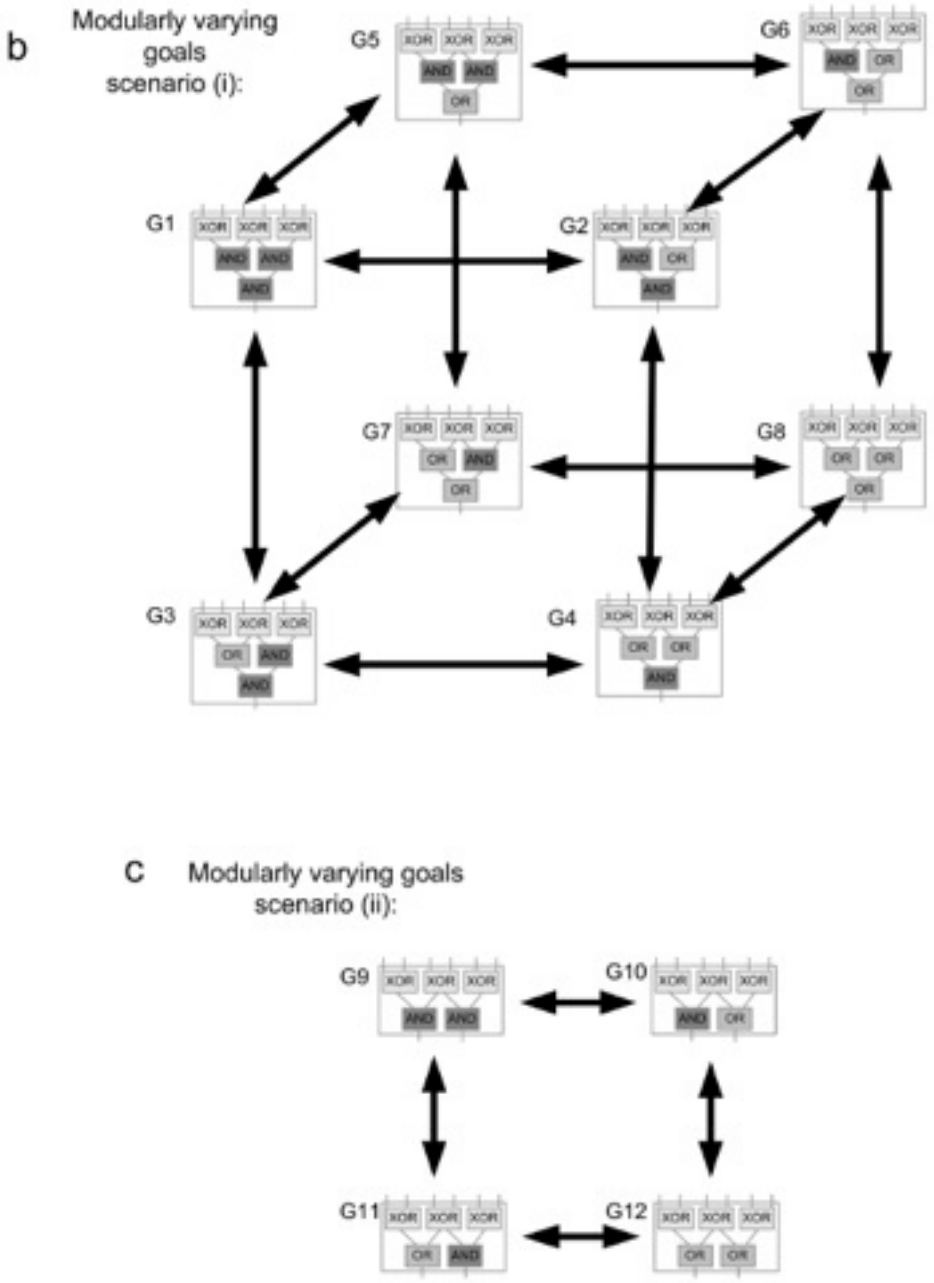


Fig. 75. MVG nel modello 1 (circuiti logici). **(a)** Scomposizione schematica del goal G1. **(b)** Scenario 1: goal con sei input e un output. Goal cambiati durante l'evoluzione in maniera probabilistica come un percorso su un grafo di 8 nodi. Per es. se il goal precedente è G1, il successivo è scelto casualmente fra uno dei tre vicini (G2, G3 o G5). Ogni modulo AND è cambiato con OR o viceversa. **(c)** Scenario 2: goal con sei input e due output. Goal cambiati durante l'evoluzione in maniera probabilistica come un percorso su un grafo di 4 nodi. Scenario 3 simile al 2 con EQ invece di XOR.

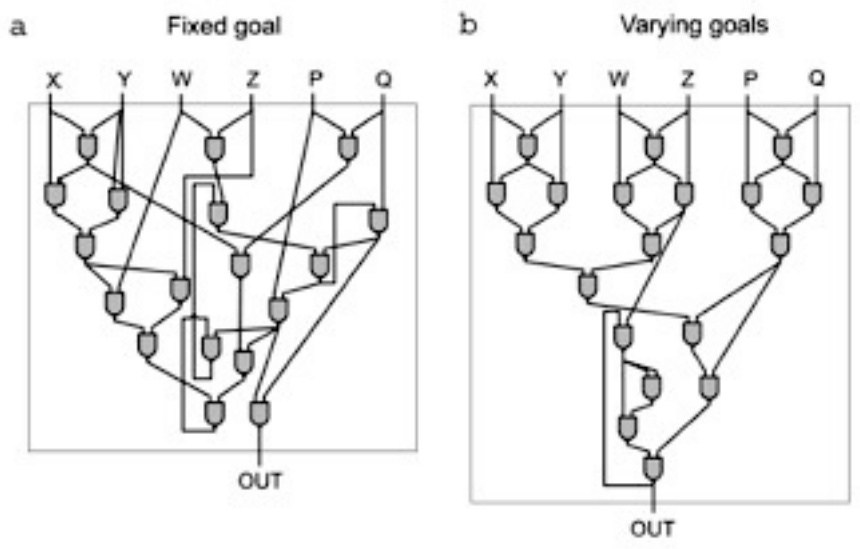


Fig. 76. Esempi di circuiti evoluti con goal fisso e MVG. **(a)** Circuito di NAND evoluti con goal fisso. Goal con sei input G1 = F(M1,M2,M3) = (M1 AND M2) AND (M2 AND M3), M1 = *x* XOR *y*, M2 = *w* XOR *z* e M3 = *p* XOR *q*. **(b)** Circuito evoluto con MVG che risolve il goal G1 con scenario 1. I circuiti evoluti con MVG hanno struttura modulare con un modulo corrispondente a ciascuno dei sottoproblemi condivisi dai goal variabili (XOR in questo caso).

I circuti logici combinatori feed-forward del modello 2 sono composti di ulteriori tipi di gate e sono costituiti di quattro layer di 8, 4, 2, 1 gate (per i goal a un output) o 3 layer di 8, 4, 2 gate (per i goal a due output); gli output sono quelli dei gate all'ultimo layer. Ogni gate può essere AND, OR o NAND. Sono consentite solo connessioni feed-forward. Goal e scenari MVG simili al modello 1.
Il genoma binario (Fig. 77) di 112 bit costituiti di 15 geni (108 bit e 14 geni per la versione con 2 output) i quali codificati per 15 gate a 2 input disposti su 4 layer con 8, 4, 2, 1 gate (3 layer con 8, 4, 2 gate per la versione con 2 output). L'output è quello del gate dell'ultimo layer. Ciascun gene è composto di tre campi: uno codificato per il tipo di gate (AND, OR o NAND) e due per le connessioni di input.

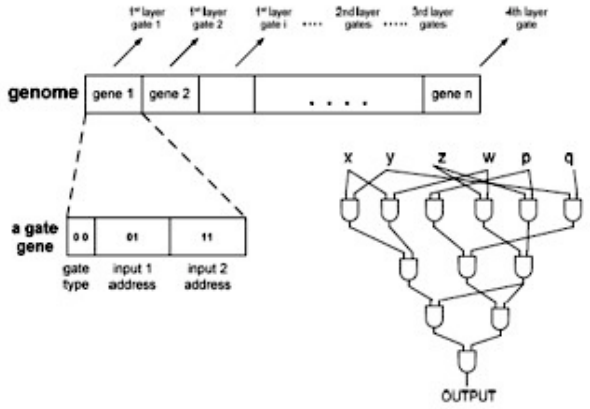


Fig. 77. Genoma binario del modello 2 e 4 di 15 geni. Ogni gene codifica per un gate. I gate disposti in feed-forward su quattro layer e il tipo di gate nel modello 2 è a 2 bit con mapping: 00: AND; 01: OR; 10: NAND; 11: AND. Il tipo di gate nel modello 4 è a 2 bit: 00 :*xy*; 01 :*x* + *y* – *xy*; 10: 1 – *xy*; 11:*xy*. Gli output come il quarto layer singolo del gate di output.

La versione ridotta del modello 2 è costituita da circuiti con tre layer di 4, 2, 1 gate; l'output del singolo gate al terzo layer e i goal a quattro input, funzioni booleane a un output.
Questo modello ha un genoma più piccolo per consentire un mapping completo del paesaggio di fitness. Il genoma binario di 38 bit, costituiti di 7 geni i quali codificati per 7 gate logici a 2 input disposti in tre layer con 4, 2, 1 gate. Connessioni solo da ogni layer al successivo e quelle di input dell'ultimo gate sono fisse (collegate agli output dei 2 gate del layer precedente).
Calcolo della fitness: goal funzioni booleane. Ogni goal definito da una tabella di verità con 4 input ($2^4 = 16$).
Il modello 3 di rete neurale *integrate-and-fire* è costituito di reti con sette neuroni disposti su tre layer feed-forward. Ciascun neurone ha due input, somma i suoi input pesati e scatta se la somma eccede una soglia. Gli input alla rete hanno tre livelli: basso, medio e alto. Il goal è identificare i pattern di input. Il goal dello scenario MVG è una combinazione R di 2 sottogoal a 2 input (S1, S2). R commutato fra AND e OR ogni 20 generazioni.

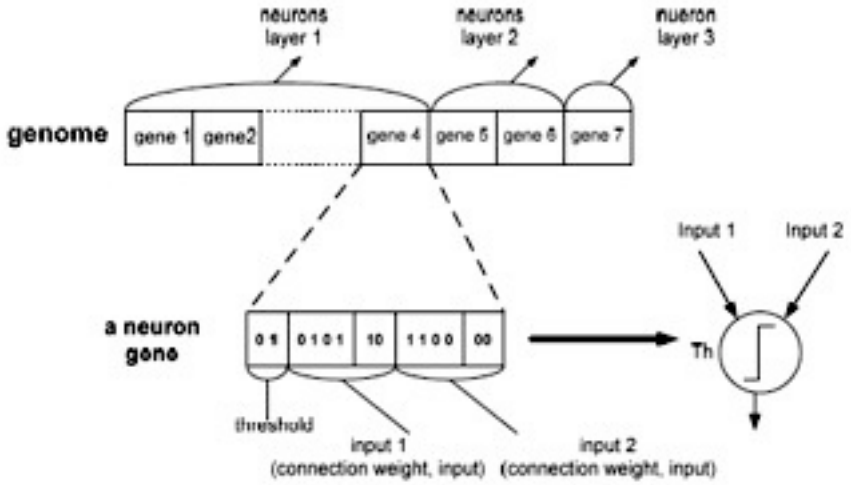


Fig. 78. Descrizione genoma modello 3. Ogni gene ha 5 campi: un campo codifica la soglia della funzione di transizione; due campi codificati per gli input usando l'indice del gate del layer precedente; due campi codificati per i pesi di connessione. Il campo soglia di 2 bit: 00: $P_{th} = -2$; 01: $P_{th} = -1$; 10: $P_{th} = 1$; 11: $P_{th} = 0$. Ogni campo peso di 3 bit: 000: W = -2; 001: W = -1.5; 010: W = -0.5; 011: W = -1; 100: W = 2; 101: W = 1.5; 110: W = 0.5; 111: W = 1. L'output è il terzo layer del neurone di output.

Calcolo della fitness: ciascuno dei quattro input della rete potrebbe essere presente in tre livelli: basso, medio e alto. Il goal è decidere se una specifica combinazione di livelli è soddisfatta e la fitness è il numero relativo di decisioni corrette su tutte le possibili combinazioni di livelli. Una rete perfetta ha fitness pari a 1.
Gli scenari *varying goal* (Fig. 79) hanno goal scelti in modo tale da essere scomposti in una combinazione di due sottogoal S1 e S2. MVG è applicata scambiando la combinazione fra R = AND e R' = OR di 8 diversi sottogoal a 2 *morphogen*. Il goal cambia ogni 20 generazioni.

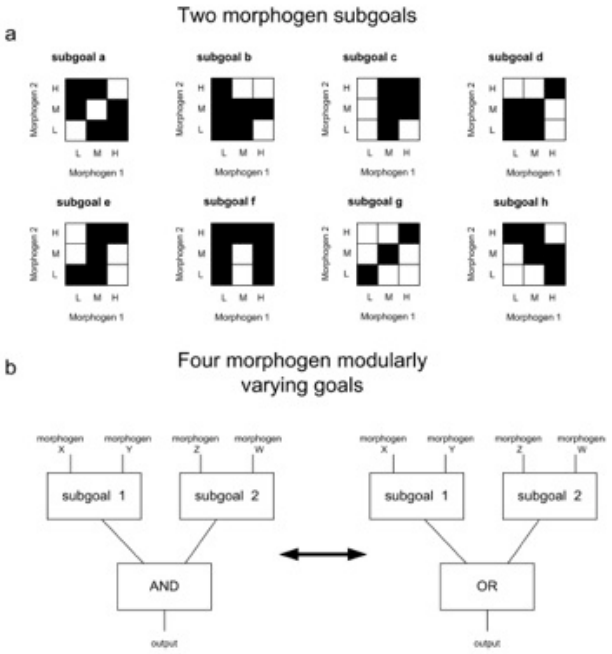


Fig. 79. *Varying goal* per il modello 3. **(a)** Ogni sottogoal ha due input (2 *morphogen*). Le concentrazioni di *morphogen* possono essere su tre livelli: basso, medio e alto rappresentato da L, M e H. Ogni sottogoal (S1, S2) identifica una combinazione unica dei due livelli di *morphogen*; (bianco = 0, nero = 1). **(b)** Goal: decidere se una combinazione di due sottogoal è identificata. Sono usate combinazioni diverse di due degli 8 sottogoal descritti in (a) dove è stata scambiata una combinazione di sottogoal R = AND e R' = OR ogni 20 generazioni.

I circuiti funzione continua del modello 4 sono costituiti di quattro layer di gate 8, 4, 2, 1 (per i goal a un output) o tre layer di 8, 4, 2 gate (per i goal a due output). Gli output quelli dei nodi all'ultimo layer.
Tre tipi di funzioni continue a due input implementano: xy, $1 - xy$, $x + y - xy$ e input con valori fra 0 e 1. Un goal definito da un polinomio multivariato di 6 variabili.
Lo scenario MVG ha goal composti di tre sottogoal (T1, T2 e T3) combinati da una funzione (U) sui loro output. T_i funzioni bilineari di una delle forme: $x + y - 2xy$ o $1 - x - y + 2xy$.
Il goal che varia cambia in maniera probabilistica ogni 20 generazioni applicando modifiche a U.
Il genoma binario di 112 bit costituito di 15 geni (108 bit e 14 geni per la versione a 2 output) i quali codificano per 15 gate di funzioni continue a due input disposti in quattro layer con 8, 4, 2, 1 gate (tre layer con 8, 4, 2 gate per la versione con due output).
L'output è quello del gate dell'ultimo layer e ogni gene è costituito di tre campi: uno per il tipo di funzione e due per gli input.
Calcolo della fitness: goal definiti come polinomi di 6 variabili x, y, w, z, p, q. Per la sua valutazione ogni input è campionato uniformemente nell'intervallo [0, 1] e la fitness definita come uno meno la distanza media relativa agli output e i valori dei goal (approssimazione dell'integrale della differenza fra la computazione della rete e la funzione goal nell'intervallo [0, 1] di tutti gli input).
Un circuito perfetto ha fitness pari 1 ma spesso non esistono circuiti perfetti quindi viene scelta una *threshold TH* pari a 0.98.
Gli scenari *varying goal* simulati sono tre:

- il goal cambia fra 8 goal con sei input e un output G1-G8 della forma:
  G = U (T1, T2,T3)

G1 = (T1 · T2) · (T2 · T3)
G2 = (T1 · T2) · [T2 + T3 - T2 · T3]
G3 = (T1 + T2 - T1 · T2) · (T2 · T3)
G4 = (T1 + T2 - T1 · T2) · (T2 + T3 - T2 · T3)

G5 = T1 · T2 + T2 · T3 - [(T1 · T2) · (T2 · T3)]
G6 = T1 · T2 + (T2 + T3 - T2 · T3) - [(T1 · T2) · (T2 + T3 - T2 · T3)]
G7 = (T1 + T2 - T1 · T2) + (T2 · T3) – [(T1 + T2 - T1 · T2) · (T2 · T3)]
G8 = (T1 + T2 - T1 · T2) + (T2 + T3 -T2 · T3) – [(T1 + T2 - T1 · T2) · (T2 + T3 -T2 · T3)]

T1 = $x + y - 2xy$
T2 = $w + z – 2wz$
T3 = $p + q – 2pq$.

Queste funzioni polinomiali rappresentano versioni continue delle funzioni XOR.

- Il goal cambia fra 4 goal a sei input e due output G9-G12 della forma:
  G = {U1 (T1,T2,T3); U2 (T1,T2,T3)}

G9 = {T1 · T2; T2 · T3}
G10 = {T1 · T2; T2 + T3 - T2 · T3}
G11 = {T1 + T2 - T1 · T2; T2 · T3}
G12 = {T1 + T2 - T1 · T2; T2 + T3 - T2 · T3}

T1 = $x + y - 2xy$
T2 = $w + z$ - 2wz
T3 = $p + q - 2pq$.

- I goal J13-J16 definiti come nella maniera precedente con:
  T1 = $1 + 2xy - x – y$
  T2 = $1+ 2wz - w - z$
  T3 = $1 + 2pq - p - q$.

Queste funzioni polinomiali rappresentano versioni continue delle funzioni EQ.

La struttura di RNA del modello 5 ha genoma di lunghezza fissa di *B* nucleotidi e ciascuna posizione potrebbe essere occupata da uno dei quattro A, U, G o C.
Calcolo della fitness: il goal è una struttura secondaria RNA e la fitness rappresenta le somiglianze della struttura attuale da quella del goal.
Come struttura prevista dalla molecola è considerata la struttura minima dell'energia libera (MFE).

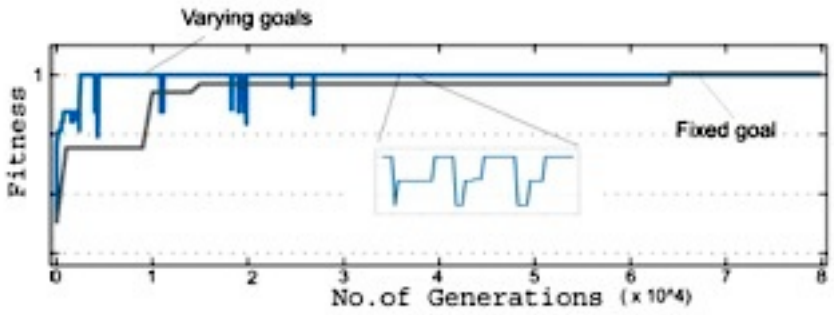


Fig. 80. Fitness con goal fisso e MVG. Grigio: la fitness massima con goal fisso; blu: fitness massima con MVG. Tempo di evoluzione: numero di generazioni di raggiungere la fitness 1 (o più grande della soglia).

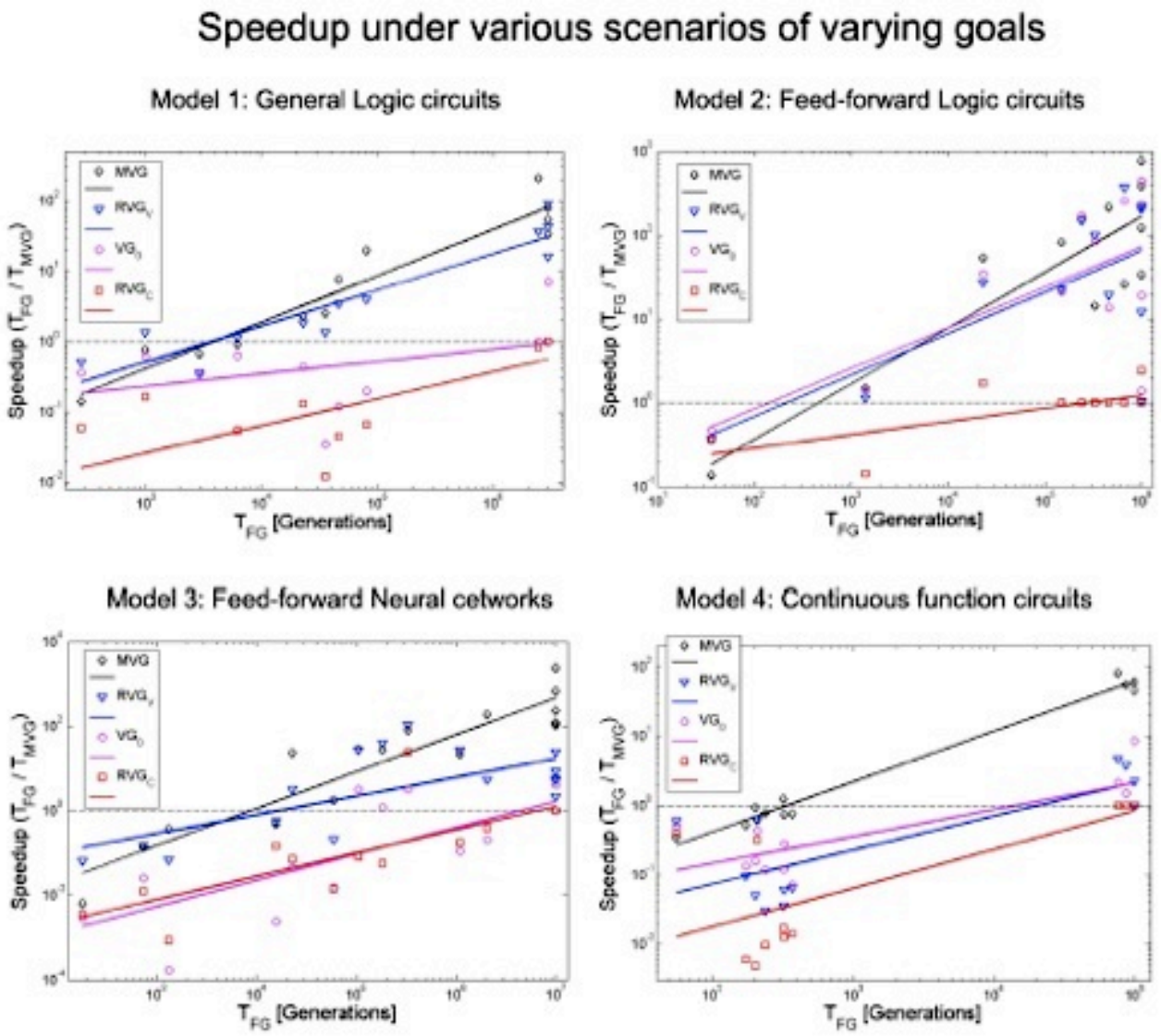


Fig. 81. Velocità di evoluzione con goal variabili. Punti e forme in colori diversi rappresentano la velocità con i quattro scenari: MVG, RVG_v, VG_0, RVG_c. Le linee di regressione calcolate usando una semplice regressione e i goal variano ogni 20 generazioni.

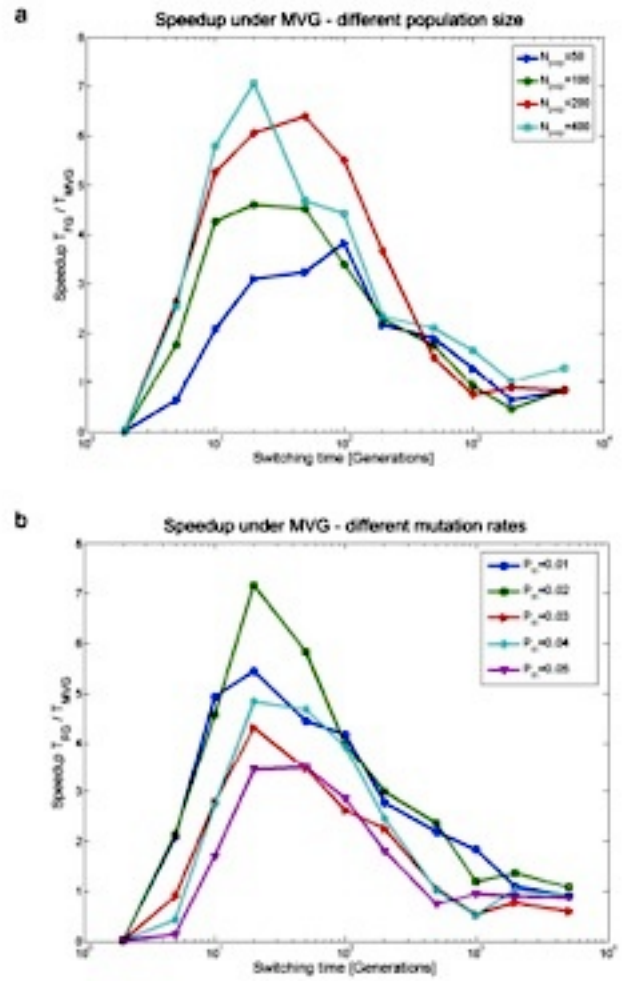


Fig. 82. Effetto dei parametri di evoluzione sulla velocità. **(a)** Velocità ($S = T_{FG}/T_{MVG}$) con diverse frequenze di *switch* di goal e con varie dimensioni di popolazione (N_{pop}). La selezione basata su fitness pura. Rate di mutazione (P_m 0.7) per genoma per generazione. **(b)** Diversi rate di mutazione (P_m è per bit per ogni generazione), $N_{pop} = 100$. Risultati per il goal G1 = (*x* XOR *y*) OR (*w* XOR *z*) con modello 2. Per MVG il goal variabile cambia fra G1 e G2 = (*x* XOR *y*) AND (*w* XOR *z*) ogni E generazioni.

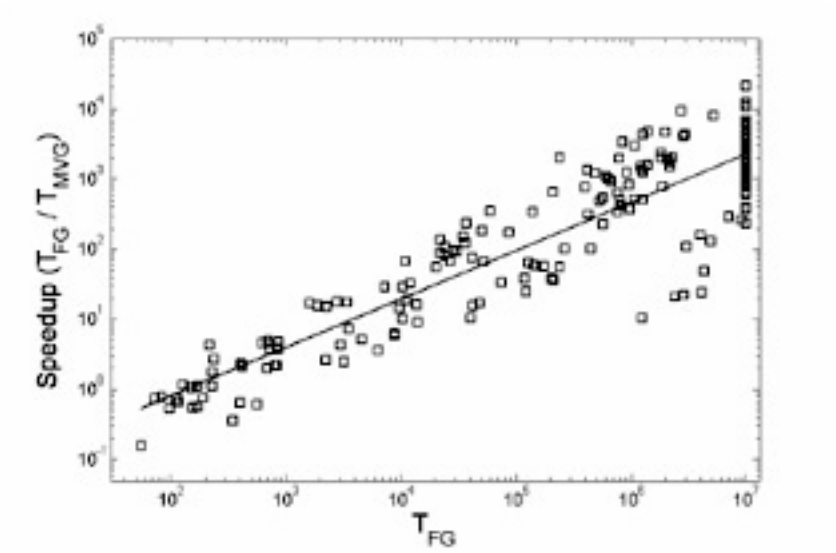


Fig. 83. Velocità con MVG con algoritmo *hill-climbing*. Ogni punto rappresenta un aumento di velocità di un goal diverso. Goal: funzioni booleane diverse a quattro input e un output (modello 2, versione ridotta). Sono scelti goal che possono essere scomposti in due funzioni booleane a due input e un output (F1, F2). Per MVG il goal cambia ogni 20 generazioni fra G1 = F1 AND F2 e G2 = F1 OR F2. La velocità diminuisce con $\alpha = 0.7$ e i risultati per $T = 0.03$.

9.4

Conseguenze computazionali delle connessioni corte

Fino ad ora abbiamo considerato schemi computazionali che consentono a un sistema di apprendimento di scoprire la struttura modulare presente nell'ambiente. Un diverso approccio che consente lo sviluppo di modularità è avere un'architettura con connessioni di piccolo raggio. Sistemi che apprendono sotto tale influenza hanno proprietà desiderabili come la scomposizione di task in sottotask, cioè si possono eliminare connessioni in modo da dedicare diverse parti del sistema a imparare task diversi; il disaccoppiamento di dinamiche di reti ricorrenti e lo sviluppo di rappresentazioni interne dipendenti dalla posizione delle unità vicine.
Quindi un vantaggio di un'architettura con connessioni corte è che può adattare la sua struttura alla natura dei task da imparare.

Conclusioni

Concludendo si può affermare che l'intelligenza artificiale si serve delle scoperte biologiche relative alla modularità per imitare gli efficienti meccanismi di apprendimento e percezione degli umani, dall'altro la biologia e la psicologia cognitiva osservano criticamente i risultati ottenuti dai computer per confrontarli con i comportamenti dei viventi e studiarli sotto una luce diversa.
In biologia, i moduli sono unità operative rilevanti. Più in dettaglio, un modulo è un'unità anatomica che si forma da una sequenza di eventi, unitari e autonomi, rispetto ad altri processi di organogenesi. Per lungo tempo l'ipotesi più in voga sulla genesi dei moduli, ha fatto leva sul concetto di adattamento secondo cui i moduli si sarebbero formati e poi trasmessi nel corso di generazioni, perché soggetti dotati di reti modulari sembravano rispondere più rapidamente all'evoluzione e quindi un maggiore adattamento. Sembra una buona spiegazione dal punto di vista dell'ereditarietà di strutture modulari, ma resta poca informazione sulla morfogenesi ed evoluzione della complessità.
I risultati di tante simulazioni hanno suggerito una maggiore efficienza funzionale di architetture modulari aiutando a spiegare la quasi universale presenza di modularità in reti biologiche anche molto diverse fra loro.
Lo scopo di questo elaborato è stato investigare sulla modularità di reti neuronali, il fatto che essa possa rappresentare un beneficio e le condizioni per consentire l'emergere dei moduli (in Appendice Tabella di sintesi).
Molti studi hanno portano a capire come e perché le reti biologiche tendono a organizzarsi in moduli, avvalendosi di un numero elevato di simulazioni sull'evoluzione al calcolatore, portando sempre più nel tempo a una comprensione dettagliata della complessità dell'evoluzione con conseguenze in *computer science* e intelligenza artificiale.
Dagli studi condotti, la modularità evolve minimizzando il costo delle connessioni fra i nodi di una rete causando la nascita di reti modulari. Massimizzare le prestazioni e minimizzare i costi produce reti più modulari e che evolvono di più rispetto a una selezione con sola prestazione.
Si è considerata l'evoluzione di reti che risolvono task di Pattern Recognition e task logici Booleani. Si è studiato il confronto fra evoluzione con goal fisso ed evoluzione con *modularly varying goal* di un circuito elettronico e una rete neurale. Risultato è stato che i secondi producono evoluzione spontanea di architetture modulari. Lo stesso processo evolutivo con goal fisso fornisce soluzioni non modulari.
Inoltre l'evoluzione con *modularly varying goal* produce reti che possono adattarsi rapidamente a ciascuno dei diversi goal solo grazie a pochi cambiamenti. L'evoluzione con goal fisso è spesso lenta e si rimane bloccati in massimi locali di fitness.
I risultati di esperimenti condotti su architetture modulari cablate e da duplicazione forniscono risultati migliori in termini di velocità di evoluzione e livello di stato stazionario che si raggiunge alla fine delle prestazioni rispetto ad architetture non modulari. Le architetture modulari originate da duplicazione tendono a favorire una specializzazione funzionale di modulo rispetto ad architetture cablate. Successivamente simulando l'evoluzione di sistemi neurali modulari, il modello "What-Where", si arriva a dire che due task indipendenti sono eseguiti in maniera più efficiente separatamente da due moduli dedicati anziché da un sistema completamente distribuito, questo per risolvere il problema dell'interferenza.
Con il modello base di rete "What-Where" costituito dall'algoritmo di apprendimento con discesa a gradiente e funzione di costo *Sum-Squared Error* si rende l'evoluzione della modularità quasi inevitabile. E' stata studiata una forma di apprendimento veloce senza eredità di pesi, arrivando cosi a rafforzare lo sviluppo di architetture modulari e una regressione accurata, con e senza eredità di pesi, la quale rafforza lo sviluppo di una configurazione modulare di pesi. Risultato è che la modularità aumenta con l'efficienza delle reti e l'aumento risulta *task-dependent*.
Un approccio diverso per consentire l'emergere della modularità è stato quello di ambienti dinamici. Essi mostrano come la struttura di una rete modulare possa essere utilizzata più

direttamente limitando l'effetto di altri parametri. Quindi ne risulta che la modularità è più visibile in ambienti dinamici.
In ultimo, si è arrivati alla conclusione che goal variabili possono accelerare l'evoluzione, in particolare esaminando i *modularly varying goal* in quattro modelli diversi. Essi velocizzano l'evoluzione e con un aumento di velocità massima che dipende dalla complessità del goal e non tutti i goal variabili mostrano aumento di velocità.
Un approccio diverso allo sviluppo della modularità interessa l'apprendimento dell'architettura, in particolare le conseguenze computazionali di un'architettura con connessioni di piccolo raggio. Sistemi che apprendono sotto tale influenza hanno proprietà desiderabili, come la scomposizione di task in sottotask, disaccoppiamento di dinamiche di reti ricorrenti e sviluppo di rappresentazioni interne *location-sensitive*.
L'obiettivo che resta ancora da capire nell'intelligenza artificiale è la creazione di reti modulari artificiali permetterà di costruire nuovi robot che possano "evolvere" acquisendo alcune caratteristiche: il loro cervello avrà un *network* di reti interconnesse in grado di riprodurre alcuni comportamenti complessi delle specie animali e quindi l'imitazione della natura permetterà l'implementazione di cervelli computazionali sempre più complessi.

Appendice: Tabella di sintesi

SIMULARE L'EVOLUZIONE DI SISTEMI NEURALI MODULARI
Bullinaria, 2001

Due task indipendenti eseguiti in maniera più efficiente separatamente da due moduli dedicati → risolta interferenza
Architetture modulari → velocità di apprendimento, interferenza spaziale e temporale, generalizzazione
Piccola connettività → apprendimento di architetture modulari

LA SPECIALIZZAZIONE PUO' GUIDARE L'EVOLUZIONE DELLA MODULARITA'
Espinosa-Soto, Wagner, 2010

Vantaggi:
Modularità → migliora capacità di adattamento; cambiamenti in un modulo senza perturbare gli altri; i moduli possono essere combinati e riutilizzati per creare nuove funzioni; partiziona le reti secondo i geni con stato di attività unico e condiviso; facilita co-opzione (combinazione di moduli evoluti precedentemente per svolgere nuove funzioni); aumenta con la selezione di un terzo pattern; facilita produzione di variazione ereditaria e innovazioni evolutive; evita di raggiungere pattern diversi selezionati in uno stesso genotipo (scenario più adatto per ambienti che non variano costantemente).

Ipotesi: scenario 1 → combinazione di selezione direzionale e selezione stabilizzante
scenario 2 → MVG

Risultati: in reti di regolazione dei geni → sottoprodotto della specializzazione dell'attività dei geni (include evoluzione di pattern di attività nuovi) → dopo specializzazione i moduli comprendono geni con cambiamenti definiti nell'attività.

EVOLUZIONE DELLA MODULARITA' TASK-DEPENDENT
Husken, Igel, Toussaint, 2002

Vantaggi:
Architetture modulari → migliore apprendimento
Modularità aumenta con l'efficienza delle reti → aumento task-dependent

Ipotesi:
due NN ottimizzate come due tipi di task → accurate model risolve un unico problema;
fast learner impara un dato problema ma in breve tempo.
Schemi di eredità: Lamarkiana con eredità di pesi;
Darwiniana senza eredità
Risultati:
Apprendimento veloce senza eredità di pesi → sviluppo di architetture modulari
Regressione accurata con o senza eredità di pesi → sviluppo di una configurazione modulare di pesi.

SCOMPOSIZIONE DI TASK ATTRAVERSO COMPETIZIONE
Jacobs, Jordan, Barto,1991

Vantaggi:
Velocità di apprendimento, crosstalk spaziale e temporale, generalizzazione di funzionalità, rappresentazione di capacità, capacità di soddisfare vincoli hardware (riduzione del numero di unità e lunghezze delle connessioni).

Risultati:
Architettura connessionistica modulare → impara a suddividere un task in due o più task funzionalmente indipendenti e assegna a ciascuno topologie di reti distinte

MODELLO DI MODULARITA' WAGNER-ALTENBERG (1996)

Evoluzione di modulo con architettura di sviluppo/genetica → evolve indipendentemente da altri moduli.

AMBIENTI FAVOREVOLI ALL'EVOLUZIONE DELLA MODULARITA'
Khare, Sendhoff e Yao, 2006

Vantaggi:
Ambienti dinamici → informazione strutturale di rete modulare meglio utilizzata e aumenta con l'aumentare del numero di sottotask all'interno del task completo.

Risultati:
due ambienti: related task;
incrementally complex task
Modularità più visibile in ambienti dinamici di quelli statici

AMBIENTI VARIABILI POSSONO ACCELERARE L'EVOLUZIONE
Kashtan, Noor e Alon, 2007

Goal variabili → accelerano l'evoluzione rispetto a goal fissi
Aumento maggiore in MVG diversi con obiettivi che cambiano, ma ciascuno condivide alcuni dei sottoproblemi con il precedente.
Velocità aumenta → complessità del goal
MVG spingono le popolazioni fuori da massimi locali di fitness → soluzioni modulari capaci di evolvere
Ambienti variabili nel tempo influenzano: struttura, robustezza, possibilità di evoluzione, mapping genotipo-fenotipo dei sistemi evoluti
Aumento di velocità in MVG e in alcune condizioni con RVG

EVOLUZIONE DI ARCHITETTURE MODULARI PER RETI NEURALI
Di Ferdinando, Calabretta, Parisi, 2000

Interferenza di reti che devono apprendere due task diversi What-Where: modularità'
Evoluzione architettura: algoritmo genetico
Apprendimento pesi: algoritmo backpropagation

CONSEGUENZE COMPUTAZIONALI DELLE CONNESSIONI CORTE
Jacobs e Jordan, 1992

Vantaggi:
Architettura con connessioni corte → scomposizione di task in sottotask, disaccoppiamento di dinamiche di reti ricorrenti, sviluppo di rappresentazioni interne dipendenti dalla posizione delle unità vicine
Adattamento della struttura alla natura dei task

EVOLUZIONE DI RETI NEURALI MODULARI ATTRAVERSO EXAPTATION
Mouret e Doncieux, 2009

Evoluzione senza collegamento diretto fra selezione e moduli → favoriti sistemi modulari
Reti monolitiche più efficienti e facili da ottenere di reti modulari

ORIGINI EVOLUTIVE DELLA MODULARITA'
Clune, Mouret e Lipson, 2013

Evolvibilità: possibilità di evoluzione → rapido adattamento a nuovi ambienti. Dovuta alla modularità

Ipotesi:
modularità evolve → selezione indiretta dell'evolvibilità; selezione diretta (minimizzare i costi di connessione fra i nodi)

Risultati:
Evoluzione di reti con selezione per sola prestazione → reti non modulari e lente ad adattarsi a nuovi ambienti. Con l'aggiunta di una selezione per minimizzare i costi → reti modulari e si adattano rapidamente a nuovi ambienti
Evoluzione in ambienti immutabili → reti non modulare e lente

INTERFERENZA GENETICA RIDUCE LA POSSIBILITA' DI EVOLUZIONE DI RETI NEURALI MODULARI
Calabretta, 2010

Vantaggi:
Architettura modulare → ciascun peso è sempre coinvolto in un unico task
Architettura non modulare → uno stesso peso può essere coinvolto in due o più task

Risultati:
Architettura modulare → ogni modulo dedicato alla soluzione di un task
Interferenza in reti non modulari che devono imparare due task contemporaneamente → più elevata nelle prime fasi dell'apprendimento
Interferenza genetica in architetture modulari e non → riduce possibilità di evoluzione
Apprendimento sequenziale in reti non modulari → task ben acquisiti e riduce interferenza

SELEZIONE NATURALE E L'ORIGINE DEI MODULI
G.P. Wagner, J. Mezey e R. Calabretta, 2005

Meccanismi di evoluzione dell'origine dei moduli:
selezione diretta per modularità → s. per evolvibilità, s. su effetti pleiotropici, modularità come fuga da vincoli della capacità di adattamento
selezione costruttiva
stabilità fenotipica
modularità che facilita adattamento fisiologico
modularità da "frustazione"
modularità come side effect dinamico

MODULARITA' O NON MODULARITA'
Bullinaria, 2002

Vantaggi:
Due task indipendenti più efficienti se eseguiti da due moduli dedicati separati invece che da un sistema omogeneo (completamente distribuito)
Possibilità di evolvere in maniera efficiente anche sistemi non modulari.
Reti modulari → rappresentazioni interne più efficienti di reti completamente distribuite e imparano più facilmente come eseguire i task
Architettura modulare con una funzione di costo adatta → migliore per perfezionare i task affidati in maniera più rapida e affidabile possibile.

Risultati:
Architettura modulare: non dipende dalla funzione di costo scelta per apprendimento.

CONNESSIONISMO EVOLUTIVO E ORIGINE DELLA MODULARITA'
Calabretta, 2002

Ipotesi:
7 modelli teorici: in tutti tranne uno, essa è direttamente o indirettamente connessa a un vantaggio selettivo

Risultati:
Modularità emerge dalla specializzazione di moduli strutturali duplicati senza alcun beneficio su prestazione ed evoluzione.
Emerge completamente come effetto collaterale di dinamiche evolutive.

LA DUPLICAZIONE FACILITA L'EVOLUZIONE DI SPECIALIZZAZIONE FUNZIONALE
Calabretta, Nolfi, Parisi e Wagner, 2000

Vantaggi:
Architetture modulari cablate e da duplicazione → migliore capacità di evoluzione e livello di stato stazionario raggiunto rispetto a quelle non modulari
Architetture modulari da duplicazione → emergere di specializzazione di moduli funzionali rispetto a quelle cablate

Bibliografia

[BAY97] Bar-Yam
"Neural Networks I: Subdivision and Hierarchy" in Dynamics of complex systems, Addison-Wesley, Cap. 2, 1997.

[BRA05] R. N. Brandon
"Evolutionary Modules: Conceptual Analyses and Empirical Hypotheses" in Modularity: Understanding the development and evolution of complex natural systems a cura di W. Callebaut, D. Rasskin-Gutman, Cambridge, The MIT Press, Cap. 3, 2005.

[BUL01] J. A. Bullinaria
"Simulating the Evolution of Modular Neural Systems", in "Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society", 2001.

[BUL07] J. A. Bullinaria
"Understanding the Advantages of Modularity in Neural Systems", in Cognitive Science, N.31, 2007.

[BUL02] J. A. Bullinaria
"To Modularize or Not To Modularize?" in Proceedings of the 2002 UK Workshop on Computational Intelligence, 2002.

[CAJ99] S. Ramòn y Cajal
"Texture of the Nervous System of Man and the Vertebrates", Nicolàs Moya, 1899.

[CAL02] R. Calabretta
"Connessionismo evolutivo e origine della modularità" in Scienze della mente, A. Borghi, T. Iacchini, Il Mulino, 2002.

[CAL05] W. Callebaut
"The Ubiquity of Modularity" in Modularity: Understanding the development and evolution of complex natural systems a cura di W. Callebaut, D. R. Gutman, Cambridge, The MIT Press, Cap. 1, 2005.

[CAL10] R. Calabretta
"Genetic interference reduces the evolvability of modular and non modular visual neural networks" in Philosophical Transactions of The Royal Society: Biological Sciences, 2010.

[CDP04] R. Calabretta, A. Di Ferdinando, D. Parisi
"Ecological neural networks for object recognition and generalization" in Neural Processing Letters, N.19, 2004.

[CDPK08] R. Calabretta, A. Di Ferdinando, D. Parisi, F. C. Keil
"How to learn multiple tasks" in Biological Theory, MIT, 2008.

[CDWP03] R. Calabretta, A. Di Ferdinando, G. P. Wagner, D. Parisi
"What does it take to evolve behaviorally complex organisms?" in Biosystems, N.69, 2003.

[CML12] J. Clune, J. B. Mouret, H. Lipson
"The evolutionary origins of modularity" in Proceedings of The Royal Society: Biological Science, N.280, 2013.

[CNPW98] R. Calabretta, S. Nolfi, D. Parisi, G. P. Wagner
"A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro- and cognitve science" in Proceedings of the Sixth International Conference on Artificial Life, 1998.

[CNPW00] R. Calabretta, S. Nolfi, D. Parisi, A. Wagner
"Duplication of modules facilitates the evolution of functional specialization" in Artificial Life, N.6, 2000.

[DCP00] A. Di Ferdinando, R. Calabretta, D. Parisi
"Evolving Modular Architectures for Neural Networks" in
Proceedings of the Sixth Neural Computation and Psychology Workshop Evolution, Learning, and Development, 2000.

[ESW10] C. Espinosa-Soto, A. Wagner
"Specialization Can Drive the Evolution of Modularity" in PLoS Comput Biol, N.6, 2010.

[HIT02] M. Husken, C. Igel, M. Toussaint
"Task-dependent evolution of modularity in neural networks" in Connection Science, N.14, Cap.13, 2002.

[JJB91] R. A. Jacobs, M. I. Jordan, A. G. Barto
"*Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks"* in Cognitive Science, N.15, 1991.

[JJO92] R. A. Jacobs, M. I. Jordan
"Computational Consequences of a Bias toward Short Connections" in Journal of Cognitive Neuroscience MIT, N.4, 1992.

[KAL05] N. Kashtan, U. Alon
"Spontaneous evolution of modulariy and network motifs" in PNAS, Vol.102, N.39, 2005.

[KNA07] N. Kashtan, E. Noor, U. Alon
"Varying environments can speed up evolution" in Proceedings of the National Academy of Sciences, 2007

[KSY06] V. R. Khare, B. Sendhoff, X. Yao
"Environments Conducive to Evolution of Modularity" in Parallel Problem Solving fron Nature – PPSN IX, Lecture Notes in Computer Science, N.4193, 2006.

[LIP07] H. Lipson
"*Principles of modularity, regularity, and hierarchy for scalable systems"* in Journal of Biological Physics and Chemistry, N.7, 2007.

[LPS01] H. Lipson, J. B. Pollack, N. P. Suh
"Promoting Modularity in Evolutionary design" in
Proceedings of DETC'01 ASME Design Engineering Technical Conferences, Pittsburgh, USA, 2001.

[MBU08] V. Landassuri-Moreno, J. A. Bullinaria
"A New Approach for Incremental Modular Neural Networks in Time Series Forecasting" in Proceedings of the Workshop on Computational Intelligence, UK, 2008.

[MDO09] J. B. Mouret, S. Doncieux
"Evolving modular neural-networks throungh exaptation" in Eleventh conference on Congress on Evolutionary Computation, 2009.

[NKA91] Natarajan, B. Kausik
"*Machine Learning – A Theoretical Approach*", Morgan Kaufmann, 1991.

[STR05] G. F. Striedter
"Principles of brain evolution", Sinauer Associates Sunderland, MA, 2005.

[WAG95] G. P. Wagner
"Adaption and the Modular Design of Organisms" in Advances in Artificial Life, 1995.

[WAG96] G. P. Wagner
"Homologues, Natural Kinds, and the Evolution of Modularity" in American Zoologist, 1996.

[WAL96] G. P.Wagner, L. Altenberg
"Complex Adaptations and the Evolution of Evolvability" in Evolution, N.50, 1996.

[WMC01] G. P. Wagner, J. Mezey, R. Calabretta
"Natural Selection and the Origin of Modules" in
Modularity. Understanding the development and evolution of complex natural systems a cura di W. Callebaut e D. Rasskin-Gutman, Cambridge, The MIT Press, 2005.

[WPC07] G. P. Wagner, M. Pavlicev, J. M. Cheverud
"The road to modularity" in Nature Reviews Genetics, N.8, 2007.