
# Modelli, sistemi e applicazioni di Vita Artificiale e Computazione Evolutiva

## WIVACE 2009

A cura di:
Orazio **Miglino**, Michela **Ponticorvo**, Angelo **Rega**, Franco **Rubinacci**

Atti del VI Workshop Italiano di Vita Artificiale e Computazione Evolutiva - Napoli 23-25 Novembre 2009

FEU

# Modelli, sistemi e applicazioni di Vita Artificiale e Computazione Evolutiva

# WIVACE 2009

A cura di
*Orazio Miglino, Michela Ponticorvo,*
*Angelo Rega, Franco Rubinacci*

Atti del VI Workshop Italiano di Vita Artificiale e
Computazione Evolutiva – Napoli 23-25 Novembre 2009

FEU

Fridericiana Editrice Universitaria
http://www.fridericiana.it/


Prima edizione italiana Novembre 2009

*Miglino, Orazio* (a cura di) :
***Modelli, sistemi e applicazioni di Vita Artificiale e Computazione Evolutiva. WIVACE 2009***/Orazio Miglino, Michela Ponticorvo, Angelo Rega, Franco Rubinacci (*a cura di*)
Napoli : Fridericiana Editrice Universitaria, 2009
ISBN 978-88-8338-092-1

1. Simulazione 2. Evoluzione 3. Intelligenza artificiale I. Titolo

*Aggiornamenti:*

15 14 13 12 11 10 09 10 9 8 7 6 5 4 3 2 1 0

# *Indice*

# *Prefazione*

WIVACE 2009, come l'edizione precedente, nasce dall'unione del Workshop Italiano di Vita Artificiale (WIVA) e della Giornata di Studio Italiana sul Calcolo Evoluzionistico (GSICE).

Dopo essere stato ospitato da Ragusa e Venezia come evento unificato, quest'anno WIVACE si tiene a Napoli, organizzato dal NAC, Laboratorio per lo studio della Cognizione Naturale ed Artificiale dell'Università di Napoli "Federico II" ed inserito nell'ambito della manifestazione di divulgazione scientifica FuturoRemoto e condivide una giornata di attività congiunte con il VI Convegno annuale dell'Associazione Italiana di Scienze Cognitive.

Nell'edizione di quest'anno è sottolineata la natura informale del convegno come luogo di incontro e officina di idee, insomma un'occasione di incontro e di discussione sul proprio lavoro per i vari gruppi di ricercatori italiani che si riconoscono nella comunità di Vita Artificiale e Computazione Evolutiva. I contributi che trovate raccolti in questo volumetto descrivono spesso dei lavori *in progress* proprio perché WIVACE 2009 si propone di accogliere questi lavori, promuoverne la discussione condivisa e accompagnarli verso sedi ambiziose come la pubblicazione su una prestigiosa rivista o una conferenza internazionale. WIVACE 2009 vuole quindi essere una palestra nella quale rafforzare i propri lavori attraverso critiche, suggerimenti, indicazioni e consigli da parte dei partecipanti così come dei revisori e soprattutto attraverso il confronto con i risultati, la metodologia e, più in generale, gli approcci di questa vivace, di nome e di fatto, comunità interdisciplinare.

La comunità interdisciplinare di WIVACE trova un punto di incontro nella condivisione dell'obiettivo scientifico della costruzione di artefatti che abbiano alcune proprietà del vivente, dalla chimica, alla biologia, alla genetica, passando per la psicologia e utilizzando gli strumenti della matematica, della fisica e dell'informatica. Inoltre è importante

sottolineare che la comunità di WIVACE è l'espressione di una comunità nazionale che ha una presenza a livello internazionale testimoniata anche dai numerosi ed importanti progetti europei di cui gli autori di cui leggerete il lavoro nelle prossime pagine sono coordinatori o partner.

Invitiamo pertanto i lettori ad approfittare del livello scientifico e dell'entusiasmo che traspira da questi lavori, dandoci appuntamento al prossimo WIVACE.

I Curatori di WIVACE 2009

# *Studi preliminari per una rete booleana capace di apprendere da esempi*

**Luca Ansaloni**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia
luca.ansaloni@unimore.it

**Marco Villani**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia.
*European Centre for Living Technology,*
S.Marco 2940 – 30124 Venezia.
marco.villani@unimore.it

**Roberto Serra**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia.
*European Centre for Living Technology,*
S.Marco 2940 – 30124 Venezia.
roberto.serra@unimore.it

## *1. Introduzione*

Le reti booleane casuali (brevemente, RBN) sono uno dei modelli più noti di sistemi complessi [1], e si sono rivelate utili per descrivere diverse importanti proprietà delle reti di regolazione genica in cellule eucariote.

Un'ipotesi di grande generalità, e particolarmente suggestiva, è quella secondo cui i sistemi soggetti a pressione evolutiva tendono a portarsi in stati particolari, spesso chiamati critici. In letteratura vi sono diverse nozioni di criticità [2, 3] fra le quali risultano particolarmente interessanti quelle legate alla dinamica. In questi casi infatti è possibile dare una precisa definizione del concetto: gli stati dinamicamente critici

sono quelli che corrispondono a regioni dello spazio dei parametri di un sistema dinamico intermedie fra quelle in cui si osserva un comportamento ordinato e quelle in cui si osservano comportamenti caotici.

In presenza di un ambiente mutevole i sistemi in stati critici presenterebbero importanti vantaggi rispetto agli altri. I sistemi caotici, infatti, sarebbero troppo suscettibili a piccole perturbazioni, mentre quelli molto ordinati sarebbero al contrario incapaci di adattarsi ai mutamenti dell'ambiente.

Diversi studi hanno evidenziato come, in funzione del valore di un parametro ben preciso, la dinamica tipica delle RBNs manifesti una transizione fra ordine e disordine, ed è stato possibile individuare con precisione la regione critica. È stato anche possibile studiare alcuni sistemi biologici, in particolare la distribuzione delle perturbazioni nei livelli di espressione genica del lievito S. cerevisiae in seguito a operazioni di knock-out di singoli geni [4, 5, 6] e l'andamento temporale delle attivazioni dei geni della linea cellulare HeLa [7], e mostrare che i sistemi studiati si comportano in maniera compatibile con l'ipotesi che si trovino in (o vicino a) stati critici.

È quindi naturale porsi la domanda se anche un sistema artificiale possa essere capace di prestazioni superiori, in un ambiente mutevole, se si trova in uno stato critico.

La letteratura su tali sistemi non ha approfondito finora questo aspetto, e il presente lavoro illustra alcuni risultati preliminari volti proprio a verificare la validità di questa ipotesi. L'architettura generale del sistema a cui pensiamo è quella di una rete booleana casuale capace di modificarsi sotto l'azione di un algoritmo evolutivo (ad esempio un algoritmo genetico) in funzione delle prestazioni su un insieme di esempi del compito da svolgere. La rete è divisa in tre parti disgiunte sebbene interagenti:

1. un insieme di nodi di input, che descrivono lo stato dell'ambiente esterno (o, se si preferisce, il pattern da riconoscere); il valore di questi nodi è prescritto dall'esterno e resta costante durante tutta l'evoluzione della rete verso un suo attrattore;
2. un insieme di nodi di output, che descrivono la risposta (diagnosi o riconoscimento) della rete;
3. un insieme di nodi interni, o nascosti.

La rete nel suo complesso è una normale rete booleana, i cui nodi interni e quelli di output cambiano nel tempo secondo una funzione di transizione booleana. La rete esegue le transizioni fino a raggiungere un attrattore e, a quel punto, l'insieme dei valori dei nodi di output

rappresenta la risposta della rete. È da definire in che modo trattare il caso in cui alcuni nodi di output siano oscillanti: è possibile utilizzare la media aritmetica dei valori oppure, se si richiede un output booleano, la regola della maggioranza.

L'apprendimento avviene in maniera perfettamente analoga a quello di una rete neurale, sulla base di un insieme di esempi (di training e di generalizzazione). La motivazione per scegliere una RBN al posto di un più familiare modello neurale consiste proprio nel fatto che per le prime è possibile definire rigorosamente il regime dinamico (ordinato, critico o caotico) della rete.

In passato vi sono stati alcuni tentativi di indurre RBN ad apprendere compiti quali l'addizione binaria [8] che hanno incontrato un successo limitato e sono stati abbandonati. Recentemente è stata proposta un'architettura simile a quella che abbiamo ipotizzato qui sopra [9].

Una caratteristica originale della nostra proposta è quella di cercare di verificare se, in presenza di un ambiente mutevole, le prestazioni di una rete critica sono superiori a quelle di reti non critiche.

A tal fine però è necessario affrontare un problema preliminare, cui è dedicato questo lavoro. Un sistema come quello sopra descritto cerca di associare ad un certo insieme di valori di input un appropriato insieme di valori di output. Come si è detto, questo richiede che la rete rilassi al suo attrattore. In generale, però, esistono diversi attrattori e quello che verrà raggiunto dipenderà dalle condizioni iniziali di tutta la rete, non solo dalla porzione di input.

Per risolvere questo problema è possibile ipotizzare di congelare i valori iniziali dei nodi interni e di output, in modo da non farli variare mentre il sistema passa da un esempio all'altro. Questo però può fortemente condizionare il comportamento della rete, dunque abbiamo iniziato uno studio del comportamento degli attrattori in funzione della dimensione relativa della porzione di input e di quelle nascoste e di output. Nel caso in cui la presenza di un input congelato fosse tale da rendere molto bassa la probabilità che il sistema possa andare verso due attrattori diversi in funzione dei valori iniziali dei nodi nascosti e di output, allora potremmo trascurare l'influenza di questi ultimi.

Si noti che, mentre il nostro studio è ispirato come si è detto dall'idea di un sistema artificiale critico capace di evolvere, l'analisi del comportamento degli attrattori in presenza di una porzione congelata della rete può essere rilevante anche per lo studio delle reti di regolazione genica: in questo caso la porzione congelata potrebbe rappresentare un insieme di geni la cui espressione è determinata per

esempio dalla presenza di particolari sostanze o dall'influenza dell'ambiente.

Nella sezione 2 viene richiamato brevemente il modello delle RBN e viene definito il nuovo formalismo, mentre nella sezione 3 vengono riportati i principali risultati e nella sezione 4 è dedicata ai commenti e indicazioni per futuri lavori.

## *2. Reti strutturate e attrattori*

Le reti booleane casuali (RBN) vengono ampiamente descritte in numerosi lavori e in questa sezione illustreremo quindi solo le nozioni principali. Per approfondire rinviamo a [1]. Una RBN è una rete composta da N nodi in cui ogni singolo nodo (o vertice) può assumere il valore 0 o 1. Il sistema viene quindi definito da un vettore i cui elementi rappresentano il valore (da qui in poi chiamato "stato") di ogni nodo.

In questo modello sia le connessioni fra i nodi che le funzioni booleane sono costanti. Lo stato del nodo i-esimo all'istante t+1 è determinato dallo stato dei suoi nodi di ingresso all'istante t e dalla propria funzione booleana. Si tratta di un sistema deterministico a stati finiti in cui si dimostra che gli stati asintotici sono cicli o eventualmente punti fissi (ovvero cicli di lunghezza 1).

Per una specifica rete, ogni nodo ha esattamente k connessioni in ingresso, provenienti da k nodi diversi scelti casualmente con probabilità uniforme fra gli altri N-1 nodi della rete. Per ogni combinazione di valori provenienti dalle connessioni in ingresso viene generata una funzione booleana assegnata al nodo $N_i$. Tale funzione avrà come output il valore 1 con probabilità p, e il valore 0 con probabilità 1-p.

In funzione del valore del cosiddetto parametro di Derrida, sono stati definiti tre regimi dinamici: reti ordinate, in cui la lunghezza dei cicli cresce lentamente col numero dei nodi, reti caotiche, in cui il periodo degli attrattori cresce esponenzialmente con il numero di nodi, e reti critiche, che costituiscono la zona di separazione dei due regimi.

Nel nostro caso le reti booleane possiedono una struttura, per cui, indicando X lo stato complessivo del sistema, con I il vettore di input, con H quello dei nodi nascosti e con O quello dei nodi di output, possiamo indicare X=[I,H,O].

I nodi di input sono trattati in maniera diversa dagli altri: i loro valori infatti sono prescritti dall'esterno (ogni esempio costituisce un particolare insieme di valori di I) mentre i nodi H e O sono modellati

normalmente. In future implementazioni i nodi di output conterranno la risposta della rete alle sollecitazioni esterne; nei successivi paragrafi siamo invece interessati al comportamento complessivo dei nodi H ed O.

## *3. Esperimenti e primi risultati*

Lo scopo delle simulazioni fin qui effettuate è quello di verificare in qual misura le RBN possiedano la capacità di discriminare fra i diversi esempi, senza essere influenzate dalle particolari condizioni iniziali possedute dai nodi nascosti e di output. Per avere tali possibilità le reti in esame devono avere le seguenti caratteristiche:

1. possedere abbastanza attrattori da poter discriminare il numero di esempi coinvolti;
2. evolvere verso lo stesso attrattore ogni volta che viene presentato lo stesso esempio.

Sono stati quindi presi considerazione 3 gruppi da 50 reti ognuno (le dimensioni dei gruppi essendo pari a 100, 500 e 1000 nodi); in ogni rete abbiamo fatto variare il numero dei nodi di input da zero fino a superare il 50% dei nodi totali. Fino a 4 nodi di input abbiamo sottoposto alla rete tutti i possibili esempi (0, 2, 4, 8, 16 esempi per rispettivamente 0, 1 , 2 , 3, 4 nodi di input); in tutti gli altri casi abbiamo mantenuto il numero di 16 esempi estratti casualmente fra tutti i possibili. Infine, per ogni esempio abbiamo fatto rilassare la rete partendo da 1000 condizioni iniziali (nel caso di 0 nodi di input abbiamo fatto comunque 1000 condizioni iniziali, come caso di riferimento).

In figura 1 è riportata la mediana del numero di attrattori per ogni differente combinazione di input, al variare della frazione dei nodi di input nella rete. Si possono fare le seguenti osservazioni:

1. RBN classiche di piccola dimensione tipicamente non possono imparare un grande numero di esempi;
2. finché la frazione di nodi di input (congelati dall' esterno) è bassa il numero tipico di attrattori diversi per ogni esempio è alto;
3. quando la frazione dei nodi di input raggiunge la metà dei nodi totali, vi è tipicamente un solo attrattore per esempio;
4. abbastanza inaspettatamente, vi è un valore del numero di nodi di input diverso da zero in cui il numero di attrattori è massimo.

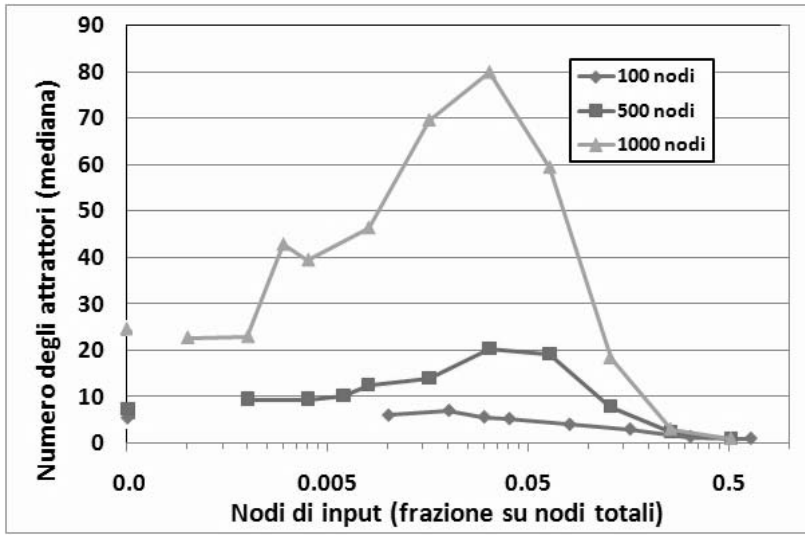



Figura 1: Numero mediano di attrattori di 3 gruppi da 50 reti ognuno, per ogni differente combinazione di input, al variare della frazione dei nodi di input della rete. I gruppi differiscono per il diverso numero di nodi (rispettivamente 100, 500 e 1000 nodi)

Interessante è la presenza del picco nel numero di attrattori corrispondenti ad ogni esempio. Una possibile causa potrebbe essere il frazionamento della rete causato dalla presenza dei nodi congelati: un numero eccessivo congela efficacemente la rete riducendone le possibilità dinamiche, mentre una piccola frazione congelata potrebbe frammentare parzialmente la rete ed influenzare in tal modo la velocità tipica di diffusione dell'informazione, permettendo quindi modalità di sincronizzazione più numerose di quelle tipiche della rete imperturbata.

Viceversa, interpretando gli esempi come stimoli dell'ambiente, la presenza di una piccola frazione di nodi mantenuti fissi dall'esterno sembra esaltare le possibilità di risposta dinamica delle RBN, offrendo quindi agli organismi la possibilità di avere/evolvere un comportamento molto flessibile.

## *4. Commenti*

Le RBNs sono sistemi complessi in cui è possibile identificare tre principali regimi dinamici: ordinato, caotico e critico. Pur non avendo finora dimostrato di poter possedere capacità di apprendimento confrontabili con quelle di altri modelli di reti neurali, le RBNs dimostrano di poter discriminare i diversi pattern loro presentati, potendo così essere utilizzate per testare l'ipotesi che sistemi critici abbiano dei vantaggi rispetto a sistemi più ordinati oppure più caotici.

In più, le simulazioni hanno mostrato un interessante picco per particolari valori del rapporto nodi I/nodi totali, che potrebbe essere associata ad un aumento di flessibilità nella risposta a stimoli esterni da parte di organismi viventi. Il fenomeno merita attenzione e sarà approfondito in ulteriori studi.

**Bibliografia**

1. Kauffman, S.A.: Origins of order. Oxford University press, Oxford, 1993
2. Bak, P., Tang, C. and Wiesenfeld, K. (1988). "Self-organized criticality". Physical Review A 38: 364–374
3. C.G. Langton, Studying artificial life with cellular automata Physica D 22 (1986) 120-140
4. Serra R., Villani M.. & Semeria A. (2004): Genetic network models and statistical properties of gene expression data in knock-out experiments Journal of Theoretical Biology 227: 149-157
5. Serra R., Villani M., Graudenzi, A., Kauffman, S.: Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. J. Theor. Biol. 246, 449-460 (2007)
6. Ramo P. Kesseli J., Yli-Harja O.: Perturbation avalanches and criticality in gene regulatory networks. J. Theor. Biol. 242, 164-170 (2006)
7. Schmulevich, A., kauffman, S.A., Aldana, M.: Eukaryotic cells are dynamically ordered or critical but not chaotic. PNAS 102, 13439-13444 (2005)
8. S. Patarnello and P. Carnevali. Learning capabilities of boolean networks. In I. Aleksander, editor, Neural Computing Architectures: The Design of Brain-Like Machines, chapter 7, pages 117-129. North Oxford Academic, London, 1989
9. Christof Teuscher, Natali Gulbahce, Thimo Rohlf Learning and Generalization in Random Boolean Networks Poster at 2007 APS March Meeting, Denver, Colorado

# *Verso l'emergere di comportamenti collettivi in robot controllati da Pattern di Turing*

**Paolo Arena**
*Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi, Università di Catania, Viale A. Doria, 6, 95125 Catania – Italy*
Tel. +39 0357382322, Fax +39 0957387985
parena@diees.unict.it

**Luca Patanè**
*Dipartimento di Ingegneria Elettrica Elettronica e dei Sistemi, Università di Catania, Viale A. Doria, 6, 95125 Catania – Italy*
Tel. +39 0357382307, Fax +39 0957387985
lpatane@diees.unict.it

Nell'ultimo ventennio si è assistito ad un interesse sempre crescente verso l'indagine di strategie utilizzate nel mondo biologico, al fine di progettare algoritmi utili nel campo della robotica mobile e dei sistemi cognitivi in generale. La nostra unità di ricerca ha preso come paradigma il mondo degli insetti, sia dal punto di vista biologico che comportamentale e strutturale. Gli insetti infatti, pur avendo dei sistemi cerebrali non troppo complessi (si parla di centinaia di migliaia di neuroni, ordini di grandezza inferiori rispetto agli animali superiori) presentano delle caratteristiche sorprendenti in termini di apprendimento "real time", adattamento ad ambienti anche estremi, robustezza e sviluppo di capacità percettive, fino a pochi anni fa ritenute presenti solo in esseri superiori, quali i vertebrati. L'aspetto sorprendente è che talune parti del cervello degli insetti, con particolare riferimento a quelle che sovrintendono compiti di apprendimento e di percezione, definizione e scelta di comportamenti, non sono note con la sufficiente accuratezza, sia a livello anatomico, che a livello funzionale. I centri principali responsabili delle capacità comportamentali e di apprendimento negli insetti sono i corpi fungiformi, noti come Mushroom bodies (MBs) ed il Complesso Centrale o Central Complex (CX). I primi sono responsabili di numerose funzioni tra cui l'apprendimento di correlazioni causali tra stimoli e reazioni, (specialmente in relazione a eventi pericolosi), im-

magazzinamento e richiamo di informazioni olfattive, risoluzione di conflitti nel caso di informazioni contraddittorie con conseguente decisione sulla scelta comportamentale. Il CX sovrintende altre classi di funzioni, tra cui l'integrazione multimodale dei comportamenti di base, l'integrazione delle informazioni visive sugli oggetti, la loro posizione nello spazio e le loro caratteristiche di interesse, il controllo diretto dell'approccio o della fuga dagli oggetti, l'orientamento.

Da quanto brevemente esposto dsi evince che il seppur "semplice" cervello di un insetto contiene al suo interno complesse strutture neurali computazionali, preziose ai fini del progetto di robot robusti ed autonomi. È quindi necessaria una stretta collaborazione tra studiosi di neurogenetica e di biorobotica per la creazione di modelli tratti da esperimenti su mutanti che possano stabilire un legame diretto tra struttura e funzione comportamentale delle parti del cervello rese funzionali su determinate linee di mutanti. I modelli ipotizzati possono essere implementati e testati successivamente su strutture robotiche, per la comparazione finale tra i risultati ottenuti con gli agenti artificiali e le prove sperimentali sui mutanti. Il fine ultimo di tale ricerca congiunta è quello di progettare e realizzare un modello computazionale del cervello degli insetti, le cui capacità cognitive possano essere comparate e validate con esperimenti mirati. Il modello, nella sua formalizzazione attuale, si basa essenzialmente su un'architettura nella quale comportamenti di base ereditati, prevalentemente connessi a modalità sensoriali singole ed utili per assolvere compiti di sopravvivenza, coesistono con strutture a livello superiore che formano in modo incrementale, a partire dalle esperienze operative dell'agente, comportamenti via via piu' complessi, grazie alla formazione di conoscenza mediata dall'ambiente circostante. A livello ancora superiore vengono formate rappresentazioni astratte delle situazioni esterne, utili per modulare i comportamenti di base al fine di soddisfare le motivazioni dell'agente stesso. La localizzazione dei siti neurali degli insetti in cui possano avvenire tali rappresentazioni è lungi dall'essere individuata, ma ai fini delle applicazioni alla robotoca mobile l'introduzione di un livello di apprendimento rappresentazionale mediato dall'ambiente ne aumenta impatto ed efficacia. L'ipotizzata struttura rappresentazionale viene modellata tramite reti neurali cellulari (CNN), progettate per l'emergere di pattern di Turing, utilizzati come stati percettivi.

Sebbene il fine preminete dell'attività di ricerca sia quello di progettare un "insect brain inspired computational model" a livello del singolo individuo, alcuni risultati preliminari lasciano intravedere come ta-

le modello possa portare all'emergere di comportamenti collettivi all'interno di una comunità di robot.

Quindi la generazione di un sistema di percezione collettiva e distribuita si presenta come il risultato ultimo ed emergente in un insieme di agenti che evolvono singolarmente, seppur soggetti a motivazioni comuni.

Questo approccio al comportamento collettivo, piuttosto che far uso, secondo i paradigmi classici degli algoritmi evoluzionistici, della trasmissione "verticale" delle caratteristiche vincenti attraverso le generazioni, utilizza una comunicazione "orizzontale" tra gli agenti che evolvono in modo concorrente ed indipendente. Il tal modo l'evoluzione ontogenetica dei singoli individui, mediata dalle esperieze ambientali, differenti per ciascu individuo, si interseca con una trasmissione orizzontale di informazioni utili per evitare situazioni negative (già sperimentate dagli altri individui) o per raggiungere più velocemente taluni obbiettivi. La comunità di robot è basata su singole capacità percettive di ciascun agente, basate sulle principali conoscenze sull'organizzazione cerebrale degli insetti, ma arricchite di rappresentazioni emergenti in CNN che generano pattern di Turing. La nascita della cooperazione si realizza quindi tramite l'emergere di "meta pattern": ciò puo' essere considerato un primitiva forma di linguaggio, un virtuoso scambio di codici (pattern) utili per l'emergere di comportamenti organizzati. In tale direzione sono stati ottenuti interessanti risultati, che tuttavia necessitano di ulteriore approfondimento.

**Bibliografia**

1. Arena, P., Fortuna, L., Lombardo, D., Patanè, L. Perception for action: Dynamic spatiotemporal patterns applied on a roving robot. Adaptve Behavior, 16(2/3), 104–121 (2008)
2. Arena, P., Crucitti, P., Fortuna, L., Frasca, M., Lombardo, D., Patanè, L.: Turing patterns in RD-CNNs for the emergence of perceptual states in roving robots. Int. Journal of Bifurcation and Chaos, 17(1), 107–127. (2007).
3. Arena, P., Patanè, L.(eds).: Spatial Temporal Patterns for Action Oriented Perception in Roving Robots. Cognitive Systems Monographs 1, Springer, ISBN: 978-3-540-88463-7. (2009).
4. Arena, P., De Fiore S., Patanè, L.: Cellular Nonlinear Networks for the emergence of perceptual states: application to robot navigation control, Neural Networks, in fase di stampa.

5. Goras L., Chua L., Turing patterns in CNNs, Part I-II, , IEEE Trans. Circuits Syst.-I, 42, pp. 602-626. (1995)
6. Murray J. D.: Mathematical Biology, Springer (2003).

# *Tecniche di Soft Computing per riconoscimento di anomalie: una prima analisi*

**Antonia Azzini**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
antonia.azzini@unimi.it

**Andrea G.B. Tettamanzi**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
andrea.tettamanzi@unimi.it

**Matteo De Felice**
*Centro Ricerche Casaccia, ENEA;*
Via Anguillarese, 301, 00060 Roma,
Tel. +39 0630484411, Fax +39 0630484811
matteo.defelice@enea.it

## *1. Introduzione*

Il riconoscimento di guasti o anomalie è uno dei processi fondamentali in diversi ambiti, legato in particolare al problema del mantenimento della sicurezza all'interno di un sistema.

Il concetto di "anomalia" di un sistema spesso parte dalla definizione di condizione di "normalità". Essa rappresenta l'obiettivo dei processi di controllo che in ogni ambito cercano di mantenere un sistema all'interno di uno spazio di funzionamento normale, con un comportamento quindi prevedibile e controllabile. Il riconoscimento automatico delle anomalie di un sistema può essere eseguito con diverse metodologie; fra esse sono interessanti quelle basate su tecniche di *machine learning*, in grado di apprendere in maniera automatica gli stati di un sistema a partire dall'osservazione dei dati.

La mancanza di una definizione formale e precisa del concetto di malfunzionamento o di anomalia e spesso la mancanza di dati utili al loro riconoscimento ha portato nel tempo ad orientarsi anche a tecniche bio-ispirate, che imitano il funzionamento dei sistemi esistenti in natura. Tra questi, uno dei più efficienti è il sistema immunitario umano, in grado di reagire a sostanze dannose sia endogene che esogene, come virus e batteri. Proprio a questo sistema sono ispirati i Sistemi Immunitari Artificiali (AIS, Hofmeyer et al. 2000), un insieme di tecniche che imitano la capacità del sistema immunitario di reagire sia ad anomalie che si presentano per la prima volta, sia ad altre già note, grazie alla capacità di apprendimento e memorizzazione delle soluzioni ai problemi già affrontati in passato. In letteratura sono stati implementati diversi algoritmi basati sugli AIS, e fra essi particolarmente interessante è l'algoritmo di Negative Selection (NS) (Hofmeyer et al., 2000).

Fra gli algoritmi che si ispirano a modelli naturali nell'ambito del riconoscimento di anomalie, risultano interessanti ed efficienti anche quelli basati sugli algoritmi evolutivi, reti neurali, e quelli basati su algoritmi di *swarm intelligence* come l'algoritmo di Particle Swarm Optimization (PSO, Kennedy et al. 1995).

A tal proposito sono state implementate e confrontate in precedenza alcune di queste tecniche nell'ambito di anomalie e guasti su reti informatiche (Azzini et al. 2009) e i risultati soddisfacenti ottenuti dalle prime analisi incoraggiano ad approfondire lo studio del comportamento degli AIS e di altre tecniche ispirate a modelli naturali nell'ambito di tali problematiche.

Il seguente lavoro si concentra sul confronto tra due delle metodologie sviluppate, cercando di apportarne dei miglioramenti anche a livello di algoritmo: un AIS basato su NS e un algoritmo che implementa un tipo di PSO.

## *2. Metodologie*

Ipersfere modellano i riconoscitori utilizzati in questo caso, e il loro volume è in grado di classificare lo spazio come anomalo o normale. Esse sono rappresentate nella seguente forma vettoriale:

$$D = [x_1 \ldots x_N \, r]$$

dove $x$ sono le coordinate nello spazio N-dimensionale e $r$ è il raggio. I principali vantaggi di tali riconoscitori si traducono nella loro facilità di

definizione e di implementazione. Un punto $z$ nello spazio viene considerato coperto se:

$$\text{dist}(D, z) < r$$

dove *dist* è una funzione di distanza o similiarità (definita in questo lavoro attraverso la funzione di distanza euclidea).

### *2.1 Negative Selection*

L'algoritmo di NS è uno dei principali algoritmi sviluppati nell'ambito degli AIS (Forrest 1994). Esso definisce un insieme di riconoscitori a copertura dello spazio complementare alla "normalità", in modo tale da classificare nuove informazioni come normali o anomale. L'algoritmo consiste nella creazione di ipersfere i cui raggi sono inizializzati casualmente nell'intervallo [0.1, 10], mentre le coordinate del centro sono inizializzate nell'intervallo [-5, 5]. Il riconoscitore appena creato viene mantenuto se non copre alcun punto definito normale. L'algoritmo termina nel momento in cui si è raggiunto il numero totale di detector richiesti.

### *2.2 Particle Swarm Optimization*

L'algoritmo PSO utilizzato per creare l'insieme dei riconoscitori (detectors) è basato su un'implementazione standard con i parametri indicati in Tabella 1. È stato implementato un sistema iterativo per la creazione dell'insieme dei detector, specificato dallo pseudocodice:

```
PUNTI_DATASET ← CARICA_DATASET()
PUNTI_NON_COPERTI ← DIMENSIONE(PUNTI_DATASET)
INSIEME_DETECTOR ← { }
WHILE (PUNTI_NON_COPERTI > 0)
    BEST_DETECTOR ← PSO_ALGORITHM(NUMERO_INDIVIDUI, NUMERO CICLI, RAGGIO MASSIMO, PUNTI_DATASET)
    PUNTI_DATASET ← PUNTIDATASET \ PUNTI_COPERTI(BEST_DETECTOR)
    PUNTI_NON_COPERTI ← DIMENSIONE(PUNTI_DATASET)
    INSIEME_DETECTOR ← INSIEME_DETECTOR AND BEST_DETECTOR
END WHILE
```

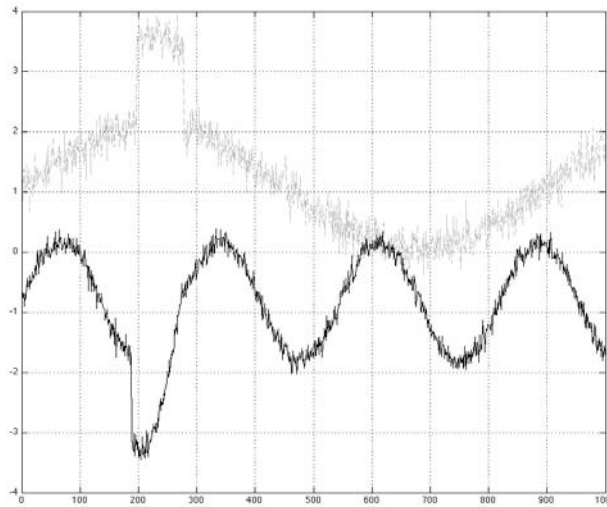



Figura 1: esempio di segnali del dataset

la funzione obiettivo dell'algoritmo di PSO implementato è basata sul numero di punti presenti all'interno del riconoscitore. Al termine di un'iterazione i punti coperti dall'individuo migliore (BEST_DETECTOR) vengono eliminati dal dataset e se questo non risulta vuoto viene eseguito di nuovo l'algoritmo di PSO sui restanti punti.

In questo modo però l'algoritmo tende a convergere ad un riconoscitore con raggio grande in modo da comprendere tutti i punti, mentre dovrebbe essere definito in modo da coprire il maggior numero di punti non-anomali, minimizzando la copertura di regioni di spazio non note. Tale problema è stato risolto inserendo un raggio massimo per i riconoscitori e un valore di penalizzazione nella funzione di fitness:

$$f(D) = \frac{\text{copertura}(D)}{(r+1)}$$

dove $D$ è il detector considerato, *copertura* è la funzione che restituisce il numero di punti coperti dal detector $D$ e $r$ è il raggio dello stesso.

### *3. Impostazioni*

I parametri utilizzati dai due modelli implementati sono definiti sulla base di una serie di esperimenti condotti al fine di individuarne il settaggio migliore. La Tabella 1 riporta le combinazioni dei valori migliori rispettivamente per PSO e NS.

Tabella 1: parametri PSO e NS

| Parametri | Valori |
|---|---|
| **PSO** | |
| $C_1$, $C_2$ | 1.49 |
| Velocità max. | 3 |
| Inerzia | descrescente da 0.95 a 0.4 |
| **NS** | |
| Raggi | [0.1,10] |
| Coordinate dei centri | [-0.5,0.5] |

Per ciascuno dei due metodi, la campagna di esperimenti è stata condotta considerando un aumento lineare della dimensionalità del problema, con valori definiti nell'intervallo [2,15].

Per ogni dimensione è stato definito un nuovo dataset. È stato creato un segnale sinusoidale con ampiezza fissa, frequenza e bias generati attraverso distribuzioni casuali uniformi. Il cambio di ampiezza presentato in un intervallo casuale del segnale indica un'anomalia. L'indice dell'anomalia di riferimento (il *target*) di un dataset a N-dim è generato dalla sovrapposizione degli intervalli di anomalia di tutti gli N segnali. Un esempio di dataset 2D è rappresentato in Figura 1.

### *4. Esperimenti e Risultati*

Diversi esperimenti sono stati condotti al variare di dim., e per ognuna di esse sono state eseguite 10 valutazioni per PSO e NS. I risultati mostrano le prestazioni ottenute all'aumentare della complessità del segnale. In particolare sono stati definiti due tipi di confronti, basati rispettivamente sul calcolo della AUC, l'area sottesa dalla curva ROC (Bradley, 1997), e sul numero di richieste della funzione di distanza uti-

lizzata nel calcolo della fitness. La Figura 2 mostra il valore di AUC delle tecniche di NS con numero diverso di detector. Da una prima analisi si osserva come l'aumento del numero di riconoscitori applicato ai segnali porta, in generale, ad una saturazione delle prestazioni degli algoritmi, dopo soddisfacenti livelli raggiunti inizialmente. Con NS si osserva infatti come un aumento dei riconoscitori (da 40000 a 100000) non migliori in modo significativo le prestazioni di alte dimensioni.

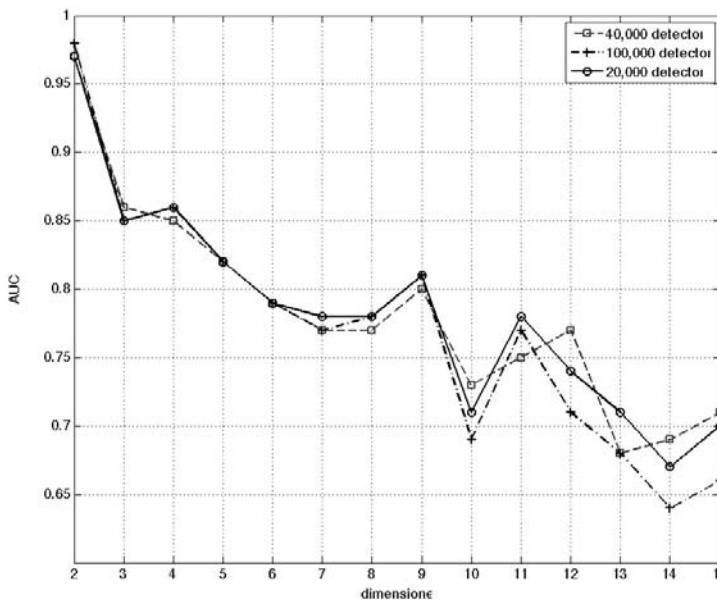



Figura 2: confronto NS con diverso numero di detector

Contemporaneamente però in entrambi i metodi si osserva anche un calo di prestazioni all'aumentare della dimensione del problema.

I migliori risultati ottenuti con PSO prevedono una popolazione formata da 100 particelle con un numero di iterazioni massimo di 200 e con un raggio definito in [1,3]. La Figura 3 mostra i risultati medi ottenuti per il PSO. Si osserva che a dimensioni elevate raggi troppo bassi non riescono a far fronte alla bassa densità dei punti nello spazio.

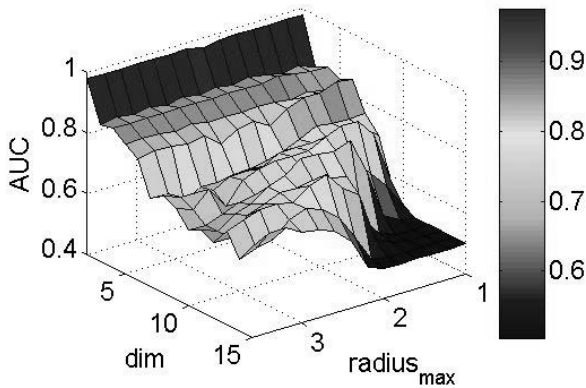



Figura 3: valore di AUC per l'algoritmo PSO al variare di dimensionalità e raggio massimo

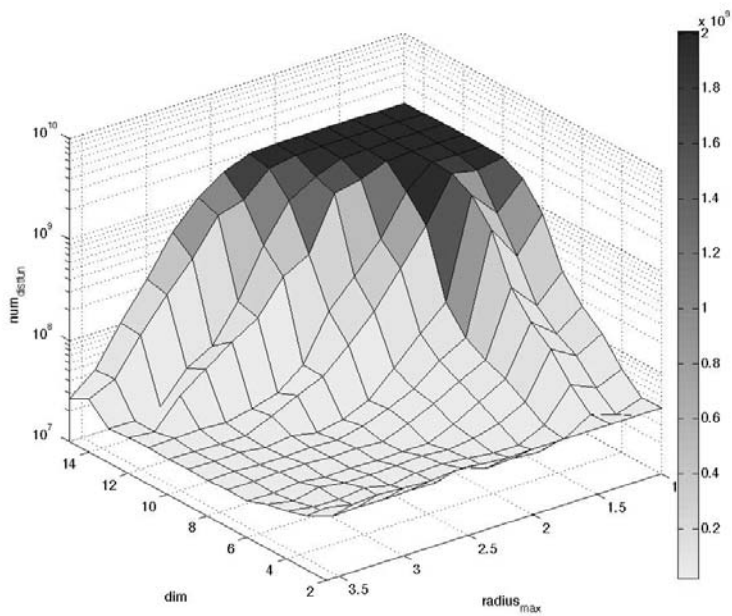



Figura 4: chiamate alla funzione di distanza del PSO

Un ulteriore ed utile strumento di confronto è indicato dallo sforzo computazionale calcolato nei vari casi, basato sul numero di chiamate alla funzione di distanza (Figura 4). Interessante notarne la crescita.

La Tabella 2 riporta un confronto fra i due metodi. Per ogni dimensione viene riportato il raggio con le migliori prestazioni.

## *5. Conclusioni*

Il riconoscimento delle anomalie è un problema poco generalizzabile e dipendente dal caso specifico considerato. Questo lavoro ha sviluppato una prima analisi, usando dataset sintetici, tra due diversi metodi, basati su tecniche di selezione opposte: una negativa, attraverso un AIS, e una positiva, con un algoritmo di *swarm intelligence*.

Le prestazioni riassunte in Tabella 2 mostrano come il PSO ottenga in generale dei risultati più soddisfacenti, ma con un aumento dello sforzo computazionale. Le Figure 3 e 4 mostrano come nel PSO prestazioni e numero di valutazioni della funzione di distanza siano correlati: un raggio minore può portare in alcuni casi ad una copertura più precisa dello spazio al costo però di uno sforzo maggiore di calcolo.

Tabella 2: confronto tra NS e PSO

| | NS – 20,000 | | NS – 40,000 | | PSO | | |
|---|---|---|---|---|---|---|---|
| Dim | AUC | N.Dist. | AUC | N.Dist. | AUC | N.Dist. | r |
| *2* | 0.97 | $2.10\ 10^7$ | 0.97 | $4.19\ 10^7$ | **0.98** | $4.46\ 10^7$ | 1 |
| *3* | **0.86** | $2.16\ 10^7$ | 0.85 | $4.32\ 10^7$ | 0.85 | $3.66\ 10^7$ | 1.2 |
| *4* | **0.86** | $2.15\ 10^7$ | **0.86** | $4.29\ 10^7$ | **0.86** | $6.76\ 10^7$ | 1.4 |
| *5* | **0.82** | $2.09\ 10^7$ | **0.82** | $4.19\ 10^7$ | **0.82** | $4.30\ 10^7$ | 1.6 |
| *6* | 0.79 | $2.09\ 10^7$ | 0.79 | $4.19\ 10^7$ | **0.84** | $5.73\ 10^7$ | 1.6 |
| *7* | 0.77 | $2.10\ 10^7$ | **0.78** | $4.20\ 10^7$ | 0.77 | $6.92\ 10^7$ | 1.6 |
| *8* | **0.78** | $2.10\ 10^7$ | **0.78** | $4.20\ 10^7$ | 0.74 | $9.12\ 10^7$ | 1.6 |
| *9* | **0.81** | $2.07\ 10^7$ | **0.81** | $4.15\ 10^7$ | **0.81** | $1.60\ 10^8$ | 1.6 |
| *10* | 0.69 | $2.06\ 10^7$ | 0.71 | $4.12\ 10^7$ | **0.74** | $8.25\ 10^7$ | 2.2 |
| *11* | 0.77 | $2.06\ 10^7$ | 0.78 | $4.11\ 10^7$ | **0.81** | $4.93\ 10^7$ | 2.4 |
| *12* | 0.71 | $2.03\ 10^7$ | 0.74 | $4.05\ 10^7$ | **0.76** | $2.94\ 10^8$ | 2.4 |
| *13* | 0.68 | $2.04\ 10^7$ | 0.71 | $4.08\ 10^7$ | **0.80** | $4.67\ 10^8$ | 2.4 |
| *14* | 0.64 | $2.02\ 10^7$ | 0.67 | $4.05\ 10^7$ | **0.74** | $3.95\ 10^8$ | 2.8 |
| *15* | 0.66 | $2.03\ 10^7$ | 0.70 | $4.06\ 10^7$ | **0.73** | $4.29\ 10^7$ | 3.4 |
| *Media* | **0.77** | $\mathbf{2.07\ 10^7}$ | **0.78** | $\mathbf{4.15\ 10^7}$ | **0.80** | $\mathbf{1.35\ 10^8}$ | |

Il PSO può quindi essere migliorato in due modi: rendendo il raggio adattivo oppure con un approccio multi-obiettivo. Il primo potrebbe essere utile per garantire una buona copertura dello spazio al crescere della dimensionalità, mentre un approccio multi-obiettivo porterebbe a massimizzare la copertura dei riconoscitori, cercando di minimizzarne il raggio. Di contro nella NS un algoritmo multi-obiettivo potrebbe cercare di trovare la copertura ottimale dei casi anomali, limitando la sovrapposizione dei riconoscitori e dei buchi che si formano nelle coperture, causa principale del calo delle prestazioni di un modello.

**Bibliografia**

1. Hofmeyer, S.A., Forrest, S.: Architecture for An Artificial Immune System, Evolutionary Computation 8(4), pp. 443-473, 2000
2. Azzini, A., De Felice, M., Meloni, S., Tettamanzi, A.G.B.: Soft Computing Techniques for Internet Backbone Traffic Anomaly Detection, Workshop on Evolutionary Applications for Telecommunications and Networks. (2009).
3. Dasgupta, D., Ji, Z.: Real-valued negative selection algorithm with variable-sized detectors. In GECCO'04. pp. 287-298, 2004.
4. Forrest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. Proc. of IEEE Symp. On Research in Security and Privacy. Pp. 202-212, Los Alamitos, CA. 1994.
5. Kennedy, J. Eberhart, R.: Particle swarm optimization. In: IEEE Interna tional Conference on Neural Networks, pp. 1942-1948 vol.4 (1995).
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition, Volume 30, Is. 7, pp. 1145-1159, 1997.

# *Una codifica lessicografica per il riconoscimento del significato delle parole utilizzando reti neurali ed algoritmi evolutivi*

**Antonia Azzini**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
antonia.azzini@unimi.it

**Célia da Costa Pereira**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
celia.pereira@unimi.it

**Mauro Dragoni**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
mauro.dragoni@unimi.it

**Andrea G.B. Tettamanzi**
*Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
Via Bramante, 65, 26013 Crema (CR)
Italy; Tel. +39 0373898025, Fax +39 0373898010
andrea.tettamanzi@unimi.it

***Abstract***

L'approccio proposto riguarda l'applicazione di un modello basato su tecniche di Soft Computing per il riconoscimento del significato corretto delle parole. Per ogni parola ambigua è stato considerato il corri-

spondente insieme dei possibili significati ed è stato utilizzato un algoritmo evolutivo per l'ottimizzazione di classificatori definiti attraverso reti neurali artificiali.

Lo schema distribuito considerato è basato su una codifica lessicografica del contesto in cui ciascuna parola è presentata.

La validità dell'approccio è dimostrata attraverso esperimenti eseguiti su un campione rappresentativo di parole ambigue.

## *1. Introduzione*

Il riconoscimento del significato di una parola (WSD), come indicato da una recente rassegna (Navigli 2009), consiste nell'assegnare il significato più adatto ad un particolare contesto ad una parola polisemica. Recenti studi in letteratura hanno presentato diversi approcci per l'automatizzazione del riconoscimento basati, in generale, su due passi: (i) considerare i possibili significati di una data parola, (ii) assegnare ogni occorrenza di una parola al suo significato più appropriato, in base sempre al contesto. La rappresentazione di quest'ultimo diventa quindi uno degli aspetti più importanti nel problema del WSD, e in letteratura sono stati presentati diverse modalità di rappresentazione, fra cui gli schemi distributiti o localizzati (Cottrell 1989, Hinton 1986).

Questo lavoro propone un nuovo schema distribuito, basato sulla codifica lessicografica per rappresentare il contesto di una data parola. Il riconoscimento del significato di una parola utilizza un approccio supervisionato basato sulla combinazione di reti neurali (NNs) e algoritmi evolutivi (EAs) (Azzini et al. 2008). L'aspetto innovativo di questo lavoro interessa l'implementazione di una nuova rappresentazione per codificare le sentenze, insieme a modifiche nell'algoritmo evolutivo.

Il progetto considera datasets di grandi dimensioni, che descrivono il contesto in cui è calato ogni significato di una parola polisemica; essi verranno utilizzati per evolvere un riconoscitore basato su NN. Viene definita quindi una classe di NNs specializzate nel riconoscimento del senso corretto della parola corrispondente.

Il contesto è definito utilizzando l'annotazione lessicografica che WordNet (C. Fellbaum 1998) assegna a ciascun significato di una parola, classificandola in una delle quarantacinque categorie basate sulla sintassi e sui raggruppamenti logici. WordNet è uno dei databases maggiormente utilizzati in letteratura in problemi legati all'analisi del testo. Esso fornisce una lista di significati per ciascuna parola contenuta, organizzati in insiemi di sinonimi (synsets).

Le attivazioni dei neuroni di ingresso delle NNs sono ottenute sommando i pattern di attivazione che codificano le parole in un dato contesto, escludendo il significato atteso (detto appunto target), rimuovendo la parola che si vuole disambiguare, le stopwords ed applicando l'algoritmo di Stemming.

Esperimenti condotti su un insieme sufficientemente rappresentativo di parole polisemiche hanno dimostrato la bontà dell'approccio. Confronti con i migliori risultati ottenuti dalle competizioni di "Semeval-1/Senseval-4" (Web ref.) hanno inoltre dimostrato come esso sia competitivo rispetto ad altre tecniche presentate in letteratura.

## *2. Approcci supervisionati per il WSD*

Interessanti aree di ricerca nel WSD riguardano l'applicazione di approcci supervisionati usati per apprendere e classificare i significati, a partire da insiemi di dati forniti in ingresso alla fase di addestramento (training) dei classificatori stessi. In questi casi i dataset impiegati devono essere sufficientemente rappresentativi per garantire soddisfacenti risultati da parte del modello implementato anche su dati precedentemente non visti, generalmente impiegati per il test e le successive validazioni.

Uno degli aspetti più critici negli approcci supervisionati riguarda la rappresentazione del contesto in cui è usata una parola. In questo approccio la rappresentazione è affidata agli ingressi delle reti neurali impiegate come classificatori. In questo lavoro è stata utilizzata una rappresentazione lessicografica per la definizione del contesto, basandosi sull'annotazione assegnata a ciascun significato da WordNet, classificando le informazioni in una delle quarantacinque categorie lessicografiche in base alla categoria sintattica e ai raggruppamenti logici. La Tabella 1 mostra un esempio di categorie lessicografiche.

Tabella 1: Esempio di Categorie Lessicografiche

| Categoria | Descrizione |
|---|---|
| noun.artifact | nouns denoting man-made objects |
| noun.location | Nouns denoting spatial position |
| noun.process | Nouns denoting natural processes |

Il contesto di ogni parola considerata *w* è rappresentato da un vettore di quarantacinque elementi, i contributi delle istanze di altre parole nelle frasi della categoria corrispondente. Tale contesto viene dato in ingresso alla rete neurale.

In generale il contributo *C_k(w)* di una istanza della parola *w* al k-esimo componente del vettore "contesto" $C$ è definito come segue:

$$C_k(w) = \frac{N_k(w)}{N(w)}$$

$N_k(w)$ è il numero di "synset" di *w* con categoria *k*, $N(w)$ è il numero di synset della parola *w*. È possibile osservare che il contributo di una parola "monosemia" è massimo, cioè pari a 1.

Sia *S* la frase in cui la parola *w* deve essere disambiguata. Il k-esimo elemento del vettore contesto *C* di *S* è dato da:

$$C_k = \sum_{w \in S} C_k(w)$$

Partendo, per esempio, dalla frase "part aqueduct system", in cui la parola "target", *tunnel*, significa (1) "un passaggio attraverso qualcosa" e (2) "un buco creato da un animale", il contributo ad ogni input della rete neurale (C1,…,C45) viene calcolato come mostra la Tabella 2. Il numero di parentesi è il numero delle istanze della corrispondente categoria lessicografica nella lista dei significati per ciascuna parola.

Tabella 2: Input della frase "part aqueduct system"

| Word | Lexicographic Category | Contribution | Word | Lexicographic Category | Contribution |
|---|---|---|---|---|---|
| part | | (18 senses) | aqueduct | | (1 sense) |
| | <noun.act> | 0.110 (2) | | <noun.artifact> | 1.000 |
| | <noun.artifact> | 0.055 (1) | | | |
| | <noun.body> | 0.055 (1) | system | | (9 senses) |
| | <noun.cognition> | 0.175 (3) | | <noun.artifact> | 0.111 (1) |
| | <noun.communication> | 0.055 (1) | | <noun.attribute> | 0.111 (1) |
| | <noun.location> | 0.055 (1) | | <noun.body> | 0.222 (2) |
| | <noun.object> | 0.055 (1) | | <noun.cognition> | 0.334 (3) |
| | <noun.possession> | 0.055 (1) | | <noun.group> | 0.111 (1) |
| | <noun.relation> | 0.055 (1) | | <noun.substance> | 0.111 (1) |
| | <verb.motion> | 0.110 (2) | | | |
| | <verb.social> | 0.055 (1) | | | |
| | <verb.contact> | 0.110 (2) | | | |
| | <adv.all> | 0.055 (1) | | | |

### *3. L'algoritmo neuro-evolutivo*

La progettazione delle reti neurali usate come classificatori è basata su un approccio evolutivo, validato su differenti benchmark e problemi reali (Azzini Tettamanzi (2008), Azzini et al. (2008)). La popolazione degli individui, i classificatori, è definita attraverso percettroni multi strato (MLPs), un tipo di NN feedforward, soggette ad addestramento, evolute attraverso l'ottimizzazione congiunta delle topologie e dei pesi.

Ogni rete è definita con un numero prestabilito di neuroni in ingresso, corrispondente alla dimensione del vettore di contesto; la dimensione dell'uscita di una rete è data dal numero di significati della parola target. Variano invece il numero di livelli intermedi di una NN e il numero di neuroni definiti in ciascun livello, per poter garantire una diversità di individui nella popolazione. Il numero di neuroni in un livello deve però essere sempre maggiore od uguale al numero di quelli del livello precedente, per evitare problemi con strutture "a clessidra", causa di diminuzione delle prestazioni.

Ad ogni generazione la metà migliore degli individui viene selezionata con un algoritmo di troncamento sul totale degli individui, viene duplicata per definirne nuovi a sostituzione dei peggiori, ed infine viene permutata casualmente. L'elitismo impedisce che l'individuo migliore in una popolazione venga cancellato di volta in volta. Successivamente, vengono mutati i pesi e la topologia degli individui della popolazione, vengono riaddestrate le reti neurali e viene calcolata infine la funzione di fitness, funzione obiettivo dell'algoritmo evolutivo, per ciascun individuo, salvandone il migliore con le informazioni statistiche dell'intero processo evolutivo. La fitness di un individuo si basa sulla matrice di confusione, in particolare sulla differenza fra il numero di neuroni di output e il parametro che corrisponde alla somma degli elementi della diagonale della matrice di confusione normalizzata per righe. La matrice di confusione è uno strumento impiegato per visualizzare il rapporto fra i casi riconosciuti in modo corretto e quelli non riconosciuti, rispetto all'insieme totale dei dati considerati, con i rispettivi valori attesi (target).

L'intero algoritmo evolutivo non usa l'operatore di ricombinazione, a causa degli effetti negativi che ne derivano (Yao, Liu (1997)).

### *4. Esperimenti e risultati*

La campagna di esperimenti è stata realizzata utilizzando gli stessi settaggi utilizzati nel precedente lavoro (Azzini, Tettamanzi (2008). I risultati degli esperimenti sono riportati nella Tabella 3. In 9 casi su 15 la codifica lessicografica migliora i risultati ottenuti con la codifica posizionale, mentre in altri 3 casi i risultati sono molto simili.

Per analizzare più in dettaglio le prestazioni del sistema vengono presentate in Figura 1 le matrici di confusione relative ai risultati ottenuti dal sistema su ciascuna delle parole analizzate.

Nei casi di *bar*, *channel*, *circuit*, *detention*, *grip*, *hearth*, *material*, *memory* e *post* l'algoritmo riesce a distinguere in modo soddisfacente la maggior parte dei significati.

Non si può dire lo stesso per le parole *bum*, *day*, *holiday* e *lady* dove i risultati ottenuti sono conseguenza della scelta dell'algoritmo neuro evolutivo di riconoscere sempre il significato più probabile.

Un caso particolare invece è dato dalla parola *dyke*, nella quale l'algoritmo riesce a distinguere con una buona precisione entrambi i significati. Questo risultato dimostra come, in presenza dataset sufficientemente rappresentativi del problema e con significati ben distinti, l'approccio presentato sia in grado di produrre soddisfacenti risultati.

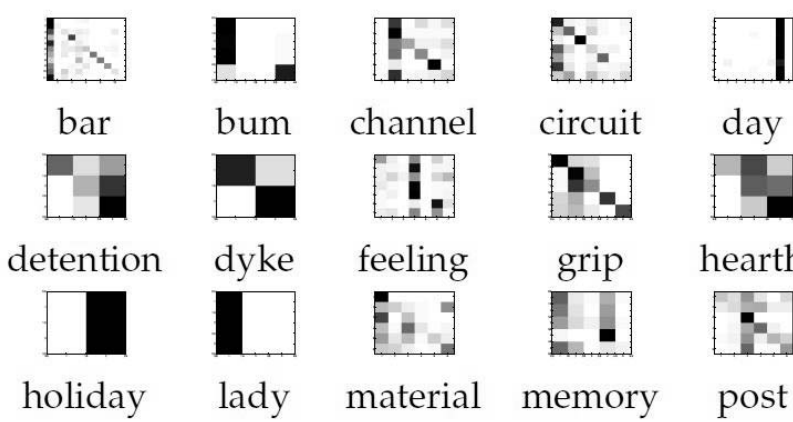



Figura 1: Matrici di confusione delle parole analizzate.

Tabella 3: Risultati ottenuti dall'algoritmo di disambiguazione e confronto con gli esperimenti condotti in precedenza (Azzini Tettamanzi 2008).

| Word | Expected Accuracy | Letter Count Accuracy | | Positional Accuracy | | Lexicographic Accuracy | |
|---|---|---|---|---|---|---|---|
| | | ANNs | EANNs | ANNs | EANNs | ANNs | EANNs |
| bar | - | 34.97 | 33.93 | 34.47 | 29.57 | 35.15 | **44.02** |
| bum | 57 | 42.39 | 42.64 | 42.45 | 47.12 | 62.62 | **63.94** |
| channel | 43 | 35.88 | 32.89 | 35.88 | 39.75 | 62.91 | **65.90** |
| circuit | 35 | 47.04 | 42.16 | 47.01 | 50.43 | 47.01 | **62.67** |
| day | 35 | 28.69 | **35.26** | 28.98 | 29.46 | 29.04 | 28.24 |
| detention | 75 | 88.25 | 92.34 | 68.68 | **95.19** | 70.18 | 71.86 |
| dyke | 75 | 96.20 | 97.40 | 91.67 | **98.73** | 95.83 | 96.38 |
| feeling | 35 | 30.26 | 32.06 | 30.26 | 35.38 | 28.16 | **40.84** |
| grip | 52 | 42.51 | 48.93 | 41.98 | 56.77 | 57.93 | **63.19** |
| hearth | 62 | 47.67 | 50.88 | 47.67 | **61.94** | 57.09 | 59.03 |
| holiday | 75 | 93.87 | 88.56 | 88.92 | 90.75 | **93.92** | 93.23 |
| lady | 62 | 45.42 | 46.61 | 45.82 | **50.15** | 45.82 | 45.82 |
| material | 43 | 68.35 | 68.36 | 68.36 | 69.73 | 68.37 | **75.62** |
| memory | 52 | 46.60 | 57.12 | 46.60 | 62.00 | 53.77 | **63.55** |
| post | 43 | 59.31 | 59.05 | 53.38 | 59.70 | 60.16 | **71.97** |

## *5. Conclusioni*

A prima vista, la creazione di una singola rete neurale per ciascuna parola ambigua potrebbe sembrare un obiettivo irrealizzabile. Tuttavia, sono presenti in WordNet solamente 15,935 parole ambigue.

Dagli esperimenti condotti risulta che la realizzazione di una rete neurale richiede circa due ore di tempo computazionale, ipotizzando quindi di utilizzare circa 30 personal computer a ciclo continuo, in circa 45 giorni tutte le parole avranno associata la loro rete neurale.

La stima di occupazione di memoria di tutte le 16.000 reti neurali è di circa 30 Mbytes e l'utilizzo della rete neurale per il corretto riconoscimento del significato di una frase richiede pochi millesimi a livello computazionale. Ciò significa che un tale approccio potrebbe essere applicato nella realtà al problema del riconoscimento dei significati corretti con risultati accettabili sia in termini di tempo che di risorse computazionali necessarie.

**Bibliografia**

1. Azzini, A., Dragoni, M., da Costa Pereira, C., Tettamanzi, A.G.B.: Evolving neural networks for word sense disambiguation. Proc. Of HIS'08, pp. 332-337, 2008.
2. Azzini, A., Tettamanzi, A.G.B.: Evolving neural networks for static single-position automated trading. J. of Artificial Evolution and Applications, pp.1-17, 2008.
3. Yao, X., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Trans. On Neural Networks, vol.8, pp. 694-713, 1997.
4. R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1-69, 2009.
5. G. Cottrell. *A Connectionist Approach to Word Sense Disambiguation*. Pitman, London, 1989.
6. G. Hinton, J. McClelland, and D. Rumelhart. Distributed representations. In *Parellel Distributed Processing: explorations in the microstructure of cognition*. MIT Press, Cambridge, MA, 1986.
7. C. Fellbaum. WordNet: An Electonic Lexical Database, MIT Press, 1998.
8. Web reference: "www.senseval.org"

# *Pianificazione automatica ottimale con ACO*

**Marco Baioletti**
*Dipartimento di Matematica e Informatica, Università di Perugia;*
Via Vanvitelli, 06123 Perugia, Italy;
Tel. +39 075-5855044, Fax +39 075-5855024
baioletti@dmi.unipg.it

**Alfredo Milani**
*Dipartimento di Matematica e Informatica, Università di Perugia;*
Via Vanvitelli, 06123 Perugia, Italy;
Tel. +39 075-5855049, Fax +39 075-5855024
milani@dmi.unipg.it

**Valentina Poggioni**
*Dipartimento di Matematica e Informatica, Università di Perugia*;
Via Vanvitelli, 06123 Perugia, Italy;
Tel. +39 075-5855045, Fax +39 075-5855024
poggioni@dmi.unipg.it

**Fabio Rossi**
*Dipartimento di Matematica e Informatica, Università di Perugia;*
Via Vanvitelli, 06123 Perugia,
Italy; Tel. +39 075-5855005, Fax +39 075-5855024
rossi@dmi.unipg.it

## *1. Introduzione*

In questo lavoro illustreremo un'applicazione dell'Ant Colony Optimization (in seguito ACO) [7] alla pianificazione automatica ottimale [9]. La pianificazione è uno dei settori principali del ragionamento automatico all'interno dell'intelligenza artificiale ed è stata studiata in maniera approfondita a partire dagli anni settanta.

L'obbiettivo della pianificazione è quello di trovare una sequenza di azioni, che, partendo da uno stato iniziale noto, permetta di raggiungere uno stato finale che soddisfa determinate condizioni goal. Ogni azione può essere eseguita solo negli stati in cui le precondizioni dell'a-

zione sono verificate e dopo la sua esecuzione lo stato corrente è aggiornato in base agli effetti dell'azione.

La pianificazione può essere vista come problema decisionale (trovare un'eventuale soluzione) o di ottimizzazione. Anche nei casi più semplici, la pianificazione è un problema computazionalmente difficile (PSPAZIO-completo), quindi è un tipo di problema in cui le tecniche stocastiche, in particolare quelle evolutive, possono essere utilizzate in maniera proficua, in mancanza di algoritmi deterministici efficienti.

Tra le varie tecniche evolutive, ACO sembra una delle più adatte al problema della pianificazione ottimale per via della similitudine tra il processo di costruzione incrementale dei piani e quello della costruzione delle soluzioni di ACO.

L'applicazione di ACO alla pianificazione presenta vari aspetti problematici, per i quali esistono svariate alternative possibili, e quindi si è resa necessaria una approfondita e prolungata serie di tentativi e di successive sperimentazioni, al fine di selezionare le migliori alternative possibili.

In questo lavoro presenteremo alcuni dei dettagli della nostra implementazione, ACOplan, insieme con i risultati sperimentali in cui confrontiamo il nostro approccio con altri pianificatori esistenti.

## *2. Pianificazione automatica*

Nel modello standard di pianificazione (modello STRIPS) il mondo è descritto mediante un insieme finito $F$ di variabili proposizionali ed uno stato generico del mondo $s$ è identificato dall'insieme delle variabili di $F$ che sono vere in $s$. Un problema di pianificazione è una terna $(I,G,A)$, in cui $I$ è lo stato iniziale, $G$ è un sottoinsieme di $F$, detto goal, e $A$ è un insieme di azioni. Ogni azione $a$ è caratterizzata da una terna $(pre(a), add(a), del(a))$ di sottoinsiemi di $F$. L'azione $a$ è eseguibile nello stato $s$ se $pre(a) \subseteq s$. Dopo l'esecuzione lo stato corrente diventa

$$Res(s,a) = s \cup add(a) - del(a)$$

Una sequenza di azioni $(a_1,a_2,\ldots,a_n)$ è un piano per un problema $(I,G,A)$ se $a_1$ è eseguibile in $s_0=I$, $a_2$ è eseguibile in $s_1 = Res(s_0, a_1)$ e così via. Un piano è una soluzione per il problema $(I,G,A)$ se $G \subseteq s_n$. Il problema di ottimizzazione prevede una funzione obiettivo $f$ definita sull'insieme delle soluzioni. In questo lavoro si userà come funzione obiettivo da minimizzare la lunghezza del piano soluzione.

Esistono svariate tecniche risolutive per la pianificazione, sia per quella classica, sia per le sue estensioni. Per maggiori dettagli si rimanda a [9].

## *3. Ant Colony Optimization*

Ant Colony Optimization (ACO) è una metaeuristica introdotta da Dorigo et al. nei primi anni '90 ispirata dal comportamento delle colonie di formiche. ACO ha lo scopo di ottimizzare una funzione *f* definita su uno spazio di sequenze finite di componenti discrete simulando una colonia di formiche virtuali. Ogni formica costruisce una soluzione scegliendo in maniera casuale una componente alla volta secondo la distribuzione di probabilità

$$p(c) = \frac{\tau(c)^{\alpha}\,\eta(c)^{\beta}}{\sum_{x}\tau(x)^{\alpha}\,\eta(x)^{\beta}}$$

in cui *c* è una generica componente, $\tau(c)$ è il valore del feromone sulla componente *c* e $\eta(c)$ è un valore euristico che indica quanto è promettente aggiungere *c* alla soluzione.

I valori di $\tau$ sono aggiornati secondo vari schemi che indicano quando e di quanto modificare il valore del feromone. In molte varianti di ACO il feromone di una componente *c* viene incrementato quando *c* fa parte di una delle migliori soluzioni, e tale incremento dipende dalla qualità della soluzione calcolata. Inoltre, in molti varianti sono previste frequenti fasi di evaporazione del feromone per evitare la convergenza della colonia verso soluzioni non ottimali. Per ulteriori dettagli si rimanda a [7].

## *4. Pianificazione con ACO*

Nella colonia ogni formica costruisce un piano, tali piani sono valutati e il feromone sarà aggiornato di conseguenza. Questo procedimento verrà ripetuto per un certo numero di iterazioni.

La costruzione del piano può avvenire sia in avanti, sia all'indietro. In questo lavoro illustreremo solo la prima modalità, in cui ogni formica parte dallo stato iniziale *I* del problema, ad ogni passo sceglie un'azione tra quelle eseguibili nello stato corrente secondo la formula *(1)* e poi

aggiorna lo stato. La costruzione termina quando lo stato corrente verifica i goal $G$, o quando non ci sono azioni eseguibili o quando si è superato la lunghezza massima consentita per i piani, fornita come ulteriore input del problema.

### *4.1 Modelli di feromone*

Dato che una soluzione di un problema di pianificazione è una sequenza di azioni, sembra naturale assegnare un valore di feromone a ciascuna azione di $A$. Ma è conveniente usare componenti che tengano conto anche del contesto in cui l'azione deve essere eseguita. In particolare si sono delineati quattro modelli di feromone, le cui componenti sono coppie del tipo

*(i)* (azione, stato corrente)
*(ii)* (azione, livello corrente)
*(iii)* (azione, livello fuzzificato)
*(iv)* (azione, azione precedente)

Il modello *(i)* è il più dispendioso in termini di numero di componenti possibili ma è anche il più informato. Il modello *(ii)* tiene conto di quanti livelli (stati intermedi) sono già stati raggiunti. Un suo miglioramento è rappresentato dal modello *(iii)* in cui i valori del feromone di un'azione $a$ ad un livello $t$ viene aggiornato, seppure in misura minore, anche se $a$ compare in livelli vicini a $t$ nelle migliori soluzioni.

Infine il modello *(iv)* tende a far scegliere le azioni in base all'azione scelta al passo precedente. Da una serie di esperimenti su varie classi di problemi è risultato che il modello *(iv)* è superiore agli altri modelli, sia in termini di risultati ottenuti, sia in termini di velocità.

### *4.2 Funzione euristica*

La funzione euristica $\eta$ gioca un ruolo essenziale nella ricerca di soluzioni, soprattutto nelle prime iterazioni, in cui raramente le formiche riescono a costruire soluzioni e quindi sono di scarso aiuto per le iterazioni successive. La funzione euristica usata in questo lavoro è $\eta_{FF}$, definita nel sistema FF [10]. Tale funzione stima la distanza, in termini di numero di azioni da eseguire, tra un qualsiasi stato ed il più vicino stato che contiene i goal.

Quindi il valore $\eta(c)$ di una componente $c$ comprendente l'azione $a$ è dato da

$$\frac{1}{h_{FF}(s_a)}$$

in cui $s_a$ rappresenta lo stato dopo l'esecuzione di $a$.

Il valore di $\eta(c)$ è moltiplicato per il fattore $1/(1-k)$ se l'azione $a$ viene classificata come *helpful* (si veda [10]).

### *4.3 Valutazione dei piani*

Uno dei problemi maggiori nell'implementazione di ACO è il fatto che la funzione obiettivo (nel nostro caso la lunghezza del piano) non è definita se il piano non è una soluzione del problema di pianificazione. Si è stabilito di usare una funzione di penalizzazione $P(\pi)$. Definendo le funzioni

- $h_{min}(\pi)=min\ \{h_{FF}(s_i) : s_i \text{ stato intermedio di } \pi\}$ che calcola la distanza minima tra gli stati intermedi di $\pi$ e i goal
- $t_{min}(\pi)=min\ \{t : h_{FF}(s_t)=h_{min}(\pi)\}$ che determina in quale momento tale distanza è raggiunta per la prima volta.

la penalità per il piano $\pi$ è data da $P(\pi)=t_{min}(\pi)+W\ h_{min}(\pi)$ in cui $W$ è un peso moltiplicativo.

### *4.4 Aggiornamento del feromone*

L'aggiornamento del feromone consiste nella valutazione di ciascuna soluzione trovata e nell'incrementare i valori di $\tau(c)$ per tutte quelle componenti che compaiono nelle migliori $\sigma-1$ soluzioni nella iterazione corrente e nella migliore soluzione fino ad allora trovata.

Se $c$ fa parte della $i$-esima migliore soluzione $\pi_{(i)}$, allora l'incremento è dato da $\Delta(c)=(\sigma+1-i)P(\pi_{(i)})$

L'evaporazione viene svolta ad ogni iterazione moltiplicando tutti i valori del feromone per il fattore $(1-\rho)$.

### *5. Risultati sperimentali*

Inizialmente sono stati fatti dei test di valutare i vari modelli di feromone proposti ed effettuare il tuning dei parametri. I migliori risultati sono stati ottenuti con il modello 4 con i seguenti parametri: 10 formiche, 5000 iterazioni, $\alpha = 2$, $\beta = 5$, $\rho = 0.15$ e $k = 0.5$.

Una seconda serie di test è stata svolta confrontando ACOplan con FF, LPG e HSP. FF [10] è stato scelto perché è basato sulla stessa euristica di ACOplan e ciò permette di valutare il contributo del framework ACO rispetto al semplice utilizzo dell'euristica. HSP [4] può essere usato anche nella versione ottimale (*-opt*). LPG [8] è un pianificatore stocastico ed è molto efficiente sia da un punto di vista computazionale che rispetto alla qualità della soluzione. Inoltre, in modalità *-n*, restituisce piani con un numero di azioni ottimale o quasi ottimale.

Tabella 1. Dominio Drivelog

| | ACOplan | | HSP | FF | HSP-opt | LPG | |
|---|---|---|---|---|---|---|---|
| **Prob** | **#az min** | **#az avg** | **#azioni** | **#azioni** | **#azioni** | **#az min** | **#az avg** |
| **drv1** | 7 | 7,00 | 7 | 8 | 7 | 7 | 7,00 |
| **drv2** | 19 | 19,15 | 24 | 22 | 19 | 19 | 19,00 |
| **drv3** | 12 | 12,00 | 14 | 12 | 12 | 12 | 12,00 |
| **drv4** | 16 | 16,15 | 21 | 16 | 16 | 16 | 16,00 |
| **drv5** | 18 | 18,20 | 22 | 22 | -- | 18 | 18,00 |
| **drv6** | 11 | 11,70 | 13 | 13 | 11 | 11 | 11,00 |
| **drv7** | 13 | 13,00 | 15 | 17 | 13 | 13 | 13,00 |
| **drv8** | 22 | 23,00 | 26 | 23 | -- | 22 | 22,00 |
| **drv9** | 22 | 22,10 | 28 | 31 | -- | 22 | 22,00 |
| **drv10** | 17 | 17,00 | 24 | 20 | -- | 17 | 17,00 |
| **drv11** | 19 | 19,25 | 24 | 25 | -- | 19 | 19,00 |
| **drv12** | 37 | 42,10 | 44 | 52 | -- | 35 | 35,25 |
| **drv13** | 26 | 26,00 | 33 | 36 | -- | 26 | 26,00 |
| **drv14** | 30 | 32,40 | 44 | 38 | -- | 28 | 28,00 |
| **drv15** | 38 | 43,15 | 48 | 48 | -- | 32 | 32,80 |

Un esempio dei risultati ottenuti sono mostrati in Tabella 1 per il dominio *Driverlog* dove abbiamo riportato la lunghezza dei piani solu-

zione trovati dai vari pianificatori e, per ACOplan e LPG, essendo pianificatori non deterministici, abbiamo riportato sia la lunghezza minima che la lunghezza media delle soluzioni, calcolate su 100 esecuzioni per ogni problema. Per gli altri domini testati i risultati ottenuti sono simili a quelli mostrati.

Dai dati in tabella è immediato vedere che ACOplan restituisce soluzioni qualitativamente molto migliori di FF e HSP. Non siamo riusciti ad ottenere un risultato significativo con HSP-opt perché riesce a risolvere solo pochissime istanze; tuttavia nei pochi casi risolti ACOplan è riuscito a trovare soluzioni ottime.

Abbiamo poi effettuato una terza fase di test per confrontare ACOplan con LPG. In Tabella 2 e Tabella 3 sono riportati alcuni dei risultati ottenuti sia in termini di qualità della soluzione che in termini di tempo di CPU per i domini *Driverlog*, *Rovers*, *Satellite*, *Openstacks*, *Parcprinter e Pegsol*. I dati riportati si riferiscono ad un sottoinsieme significativo dei problemi contenuti nei benchmark. Poiché entrambi i sistemi sono non deterministici abbiamo riportato i valori medi calcolati su 100 esecuzioni.

Tabella 2: Domini Drivelog e Pegsol

| | Min Length | | Avg Length | | Avg CPUTime | |
|---|---|---|---|---|---|---|
| **Problem** | **ACO** | **LPG** | **ACO** | **LPG** | **ACO** | **LPG** |
| **drv1** | 7 | 7 | 7,00 | 7,00 | 0,01 | 0,03 |
| **drv2** | 19 | 19 | 19,15 | 19,00 | 6,49 | 0,53 |
| **drv3** | 12 | 12 | 12,00 | 12,00 | 0,03 | 0,08 |
| **drv4** | 16 | 16 | 16,15 | 16,00 | 9,9 | 0,08 |
| **drv5** | 18 | 18 | 18,20 | 18,00 | 39,81 | 2,28 |
| **drv6** | 11 | 11 | 11,70 | 11,00 | 21,37 | 0,85 |
| **drv7** | 13 | 13 | 13,00 | 13,00 | 0,27 | 0,08 |
| **drv8** | 22 | 22 | 23,00 | 22,00 | 39,52 | 9,25 |
| **drv9** | 22 | 22 | 22,10 | 22,00 | 24,03 | 0,54 |
| **drv10** | 17 | 17 | 17,00 | 17,00 | 8,22 | 0,54 |
| **drv11** | 19 | 19 | 19,25 | 19,00 | 14,16 | 23,06 |
| **drv12** | 37 | 35 | 42,10 | 35,25 | 615,58 | 237,28 |
| **drv13** | 26 | 26 | 26,00 | 26,00 | 39,81 | 4,55 |
| **drv14** | 30 | 28 | 32,40 | 28,00 | 152,42 | 12,54 |
| **drv15** | 38 | 32 | 43,15 | 32,80 | 2504,1 | 499,78 |

| | Min Length | | Avg Length | | Avg CPUTime | |
|---|---|---|---|---|---|---|
| **Problem** | **ACO** | **LPG** | **ACO** | **LPG** | **ACO** | **LPG** |
| **peg01** | 5 | 5 | 5 | 5 | 0,33 | 0,01 |
| **peg02** | 9 | 9 | 9 | 9 | 0,33 | 0,79 |
| **peg03** | 9 | 9 | 9 | 9 | 0,34 | 1,09 |
| **peg04** | 10 | 10 | 10 | 10 | 0,57 | 0,02 |
| **peg05** | 11 | 11 | 11 | 11 | 0,4 | 4 |
| **peg06** | 12 | 12 | 12 | 12,45 | 0,53 | 29,33 |
| **peg07** | 12 | 12 | 12 | 12 | 0,4 | 0,11 |
| **peg08** | 16 | 16 | 16,15 | 19 | 4,2 | 44,96 |
| **peg09** | 15 | 15 | 15 | 16,4 | 1,26 | 8,89 |
| **peg10** | 17 | 17 | 17,85 | 17,65 | 4,05 | 52,81 |
| **peg11** | 18 | 18 | 18 | 19,2 | 1,2 | 32,38 |
| **peg12** | 20 | 20 | 20 | 22,25 | 1,37 | 41,65 |
| **peg13** | 21 | 22 | 21,85 | 22,8 | 3,05 | 31,74 |
| **peg14** | 20 | 20 | 20,05 | 20,25 | 5,83 | 63,29 |
| **peg15** | 21 | 22 | 21,95 | 23,75 | 4,13 | 30,69 |

Tabella 3: Domini Rover e Openstacks

| | Min Length | | Avg Length | | Avg CPUTime | |
|---|---|---|---|---|---|---|
| **Problem** | **ACO** | **LPG** | **ACO** | **LPG** | **ACO** | **LPG** |
| **rvr1** | 10 | 10 | 10 | 10 | 0,01 | 0,04 |
| **rvr2** | 8 | 8 | 8 | 8 | 0,01 | 0,01 |
| **rvr3** | 11 | 11 | 11 | 11 | 0,3 | 0,71 |
| **rvr4** | 8 | 8 | 8 | 8 | 0,01 | 0,02 |
| **rvr5** | 22 | 22 | 22 | 22 | 0,11 | 0,04 |
| **rvr6** | 36 | 36 | 36 | 36 | 3,33 | 2,32 |
| **rvr7** | 18 | 18 | 18 | 18 | 0,07 | 10,86 |
| **rvr8** | 26 | 26 | 26 | 26,1 | 0,28 | 158,48 |
| **rvr9** | 31 | 31 | 31 | 31 | 7,07 | 0,09 |
| **rvr10** | 35 | 35 | 35 | 35 | 9,13 | 133,03 |
| **rvr11** | 30 | 30 | 30,65 | 30 | 222,39 | 13,41 |
| **rvr12** | 19 | 19 | 19 | 19 | 0,16 | 0,17 |
| **rvr13** | 43 | 43 | 43,95 | 43,65 | 52,95 | 190,79 |
| **rvr14** | 28 | 28 | 28 | 28 | 22,07 | 0,94 |
| **rvr15** | 41 | 42 | 41,15 | 42,25 | 622,45 | 474,17 |

| | Min Length | | Avg Length | | Avg CPUTime | |
|---|---|---|---|---|---|---|
| **Problem** | **ACO** | **LPG** | **ACO** | **LPG** | **ACO** | **LPG** |
| **opn1** | 17 | 17 | 17,00 | 17,00 | 0,07 | 0,96 |
| **opn2** | 20 | 20 | 20,00 | 20,00 | 0,13 | 1,15 |
| **opn3** | 23 | 23 | 23,00 | 23,00 | 0,26 | 6,53 |
| **opn4** | 27 | 27 | 27,00 | 27,10 | 0,29 | 7,18 |
| **opn5** | 31 | 31 | 31,00 | 31,00 | 0,96 | 2,07 |
| **opn6** | 32 | 32 | 32,00 | 32,95 | 40,78 | 34,65 |
| **opn7** | 38 | 38 | 38,00 | 38,00 | 0,78 | 32,25 |
| **opn8** | 41 | 41 | 41,00 | 41,05 | 1,84 | 66,96 |
| **opn9** | 42 | 42 | 42,67 | 43,10 | 93,09 | 92,49 |
| **opn10** | 46 | 46 | 46,38 | 48,57 | 114,57 | 40,91 |
| **opn11** | 49 | 49 | 49,67 | 50,51 | 109,91 | 70,79 |
| **opn12** | 53 | 53 | 53,00 | 54,00 | 75,96 | 136,65 |
| **opn13** | 57 | 57 | 57,00 | 57,60 | 27,11 | 118,62 |
| **opn14** | 60 | 60 | 60,20 | 62,20 | 198,01 | 108,66 |
| **opn15** | 63 | 63 | 63,40 | 64,40 | 45,94 | 224,46 |

Come spesso riportato in letteratura, le prestazioni di ogni pianificatore dipendono fortemente dal dominio in uso e, anche all'interno dello stesso dominio, i risultati possono variare significativamente.

In Figura 1 abbiamo invece riassunto i risultati ottenuti per risolvere tutti i problemi. Nei grafici i colori rappresentano i tre possibili esiti del confronto tra ACOplan e LPG. Da tale figura si evince che i due sistemi sono confrontabili, almeno per la qualità delle soluzioni prodotte.

## *6. Conclusioni e sviluppi futuri*

In questo lavoro è stata presentata un'applicazione delle tecniche ACO alla pianificazione automatica per risolvere problemi di pianificazione ottimale rispetto alla lunghezza dei piani soluzione. I risultati dei test svolti finora sono incoraggianti e mostrano come questo approccio possa portare ulteriori miglioramenti nel settore della pianificazione automatica. Tra i possibili sviluppi futuri citiamo

- ulteriori esperimenti per poter confrontare i vari modelli di feromone e le due direzioni (avanti e indietro)

- estendere l'approccio a problemi in cui le azioni hanno un costo, così da poter confrontare ACOplan con i più recenti pianificatori presentati all'ultima planning competition [5].
- applicare le tecniche ACO in modelli di pianificazione estesi, quali la pianificazione con risorse numeriche, con preferenze o la pianificazione temporale
- studiare l'andamento delle soluzioni in termini di convergenza verso la soluzione ottima.

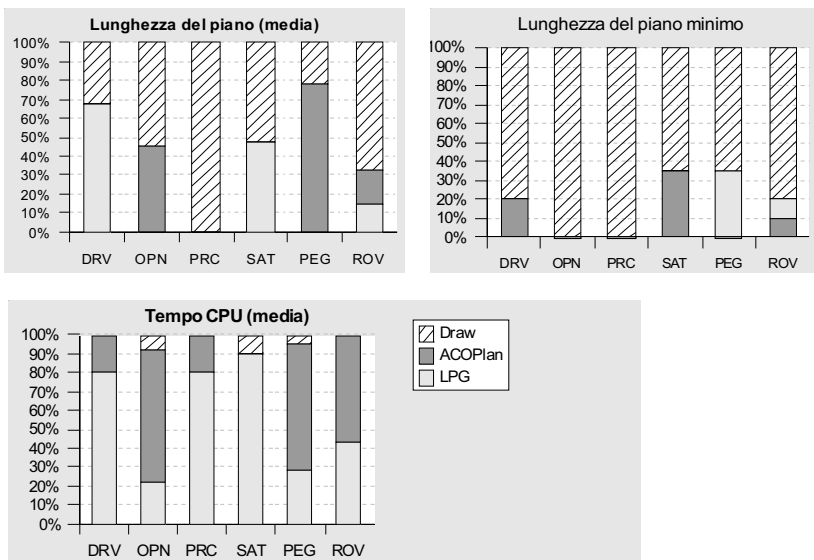



Figura 1

## Bibliografia

1. M. Baioletti, A. Milani, V. Poggioni, F. Rossi: An ACO approach to planning. In Proc of EVOCOP 2009
2. M. Baioletti, A. Milani, V .Poggioni, F. Rossi: Ant search strategies for planning optimization. In Proc. of ICAPS 09

3. M. Baioletti, A. Milani, V. Poggioni, F. Rossi: PlACO: Planning with Ants. In Proc of FLAIRS 2009
4. P. Haslum, B. Bonet, H. Geffner: New Admissible Heuristics for Domain-Independent Planning, In Proc of AAAI 2005
5. M. Helmert, M. Do, I. Refanidis, International Planning Competition IPC-2008, The Deterministic Part. http://ipc.icaps-conference.org/
6. M. Dorigo, L.M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. In "IEEE Transactions on Evolutionary Computation", 1(1), 53-66, 1997
7. M. Dorigo, T. Stützle: Ant Colony Optimization, MIT Press, Cambridge, 2002
8. A. Gerevini, I. Serina, LPG: a Planner based on Local Search for Planning Graphs, In Proc of AIPS 2002
9. D. Nau, M. Ghallab, P. Traverso: Automated Planning: Theory and Practice, Morgan Kaufmann, 2004
10. J. Hoffmann, B. Nebel: The FF Planning System: Fast Plan Generation Through Heuristic Search, In JAIR, 14, 253-302, 2001

# *Il Progetto IM-CLeVeR: Intrinsically Motivated Cumulative Learning Versatile Robots*

**Gianluca Baldassarre**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia 44, 00185 Roma
Tel. +39 06 44595231, Fax +39 06 44595243
gianluca.baldassarre@istc.cnr.it

**Marco Mirolli**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia 44, 00185 Roma
Tel. +39 06 44595231, Fax +39 06 44595243
marco.mirolli@istc.cnr.it

## *1. Introduzione*

Questo abstract esteso presenta le idee chiave del progetto IM-CLeVeR. IM-CLeVeR è un progetto finanziato dall'Unione Europea (finanziamento europeo: 5.899.884 euro), di durata quadriennale (dall'1 Gennaio 2009 al 30 Aprile 2013), e con consorzio di 7 partners europei coordinato dal "LOCEN – Laboratory of Computational Neuroscience" dell' ISTC-CNR di Roma. Il progetto mira a sviluppare una nuova metodologia per il design di controllori di robot che: (a) apprendono nuove abilità in modo cumulativo attraverso lo sviluppo autonomo basato su motivazioni intrinseche, e (b) sanno riusare tali abilità per realizzare compiti assegnati loro esternamente. Questo obiettivo sarà perseguito investigando tre argomenti fondamentali: (a) i meccanismi di astrazione dell'informazione sensoriale; (b) i meccanismi sottostanti alle motivazioni intrinseche; (c) le architetture gerarchiche che consentono l'apprendimento cumulativo. Lo studio di questi argomenti sarà condotto sulla base di esperimenti empirici condotti con scimmie, bambini e adulti umani attraverso modelli biologicamente vincolati diretti a riprodurre ed interpretare i risultati di tali esperimenti ed attraverso la progettazione, implementazione e test di metodi innovativi di machine learning. I modelli, le architetture, e gli algoritmi così sviluppati saran-

no validati con esperimenti e dimostratori condotti con il robot umanoide iCub sia in simulato che reale.

## *2. Gli obiettivi generali del progetto*

Come possiamo creare robot veramente intelligenti? Questo obiettivo ha un'importanza tecnologica e scientifica enorme. Come tecnologia, i robot intelligenti possono essere utilizzati per migliorare la qualità della vita umana, per esempio per eseguire compiti ripetitivi o condurre missioni in ambienti ostili, ad esempio nei fondali marini, nello spazio, in regioni altamente inquinate o radioattive. Dal punto di vista scientifico, la capacità di costruire robot veramente intelligenti può aiutare a scoprire i meccanismi fondamentali di funzionamento del cervello, sottostanti l'apprendimento ed l'intelligenza umana. A parte il valore di questo nel lungo termine per la tecnologia, una migliore comprensione del cervello umano può consentire un migliore trattamento delle malattie psichiatriche e neurologiche.

Il progetto IM-CLeVeR mira a sviluppare una nuova metodologia per costruire robot intelligenti basati sull'apprendimento cumulativo di abilità basato sulle motivazioni intrinseche. L'idea centrale dietro questa nuova metodologia è che, invece di programmare direttamente o addestrare o evolvere uno specifico set di abilità nei robot, essi dovrebbero essere dotati di programmi di sviluppo che consentano loro di sviluppare autonomamente le abilità di cui hanno bisogno sulla base di lunghi periodi di interazione con l'ambiente guidati da motivazioni intrinseche. I robot dovrebbero poi essere in grado di usare le abilità generali così ottenute come base per sviluppare soluzioni per i compiti utili agli utilizzatori dei robot stessi.

Si noti come queste capacità segnano alcuni dei più intelligenti aspetti del comportamento degli organismi complessi, in particolare gli umani ed i primati non umani. Per esempio, i bambini che giocano conducono un'attività incessante guidata solo da motivazioni intrinseche come la curiosità. Queste attività consentono loro di acquisire conoscenza ed abilità poi sfruttate per perseguire obiettivi utili in età adulta.

### *3. Temi di ricerca del progetto*

L'ipotesi centrale del progetto è che l'apprendimento cumulativo dei robot deve basarsi su tre principi fondamentali:

*Architetture gerarchiche*. Le architetture cognitive dei robot dovrebbero avere la capacità di sviluppare capacità cognitive e sensomotorie in un modo incrementale e gerarchico. Qusto richiede: (a) l'acquisizione di abilità e l'incremento sistematico della loro complessità; (b) l'apprendimento di nuove abilità sulla base di abilità apprese in precedenza; (c) l'immagazzinamento di nuove abilità senza dimenticare (e possibilmente migliorando) quelle acquisite in precedenza.

*Rilevamento di novità e motivazioni intrinseche*. I robot che apprendono cumulativamente hanno bisogno di motivazioni interne che focalizzano l'apprendimento sulle abilità che: (a) sono nuove per i robot; (b) sono entro la "zona di sviluppo prossimale" (Vygotsky, 1978), cioè i robot debbono avere la motivazione ad acquisire nuove abilità che possono effettivamente essere acquisite sulla base di quelle già acquisite. Per far questo, i robot dovrebbero essere dotati di "motivazioni intrinseche" che li inducono a condurre attività che producono il massimo tasso di apprendimento e di accumulo di informazioni. Le motivazioni intrinseche differiscono dalle "motivazioni estrinseche" in quanto queste ultime sono associate ai risultati pratici prodotti dalle azioni sul mondo (per esempio, l'ottenimento di cibo o sesso negli organismi, gli obiettivi dell'utilizzatore nei robot). L'apprendimento guidato da motivazioni intrinseche deve basarsi su "rilevatori di novità", ovvero meccanismi capaci di monitorare e misurare il livello di novità soggettiva dei risultati delle azioni ed i tassi di apprendimento così da focalizzare il sistema su esperienze che massimizzano il suo tasso di apprendimento.

*Astrazione sensoriale ed attenzione*. Benché l'astrazione sensoriale è un argomento ampiamente investigato nelle scienze cognitive (per esempio nella computer vision), il progetto mira ad isolare e studiare i problemi peculiari dell'astrazione legati ai temi del progetto, ovvero alla rilevazione di novità e alle architetture gerarchiche per l'apprendimento cumulativo.

### *4. Obiettivi specifici*

M-CLeVeR ha quattro obiettivi scientifici e tecnologici principali:

*Avanzare la nostra conoscenza sull'apprendimento cumulativo negli organismi.* Per far questo, il progetto prevede la conduzione di esperi-

menti non invasivi sull'apprendimento con motivazioni intrinseche nelle scimmie, nei bambini, negli adulti umani, e nei malati di Parkinson, sulla base di due nuovi paradigmi sperimentali.

*Sviluppare modelli bio-vincolati* (sia modelli simulativi che modelli robotici) con l'obiettivo di riprodurre e spiegare i risultati empirici ottenuti con gli esperimenti del punto precedente. Accanto al suo valore scientifico, questo obiettivo consentirà di isolare nuovi principi computazionali utilizzabili nei robot.

*Sviluppare nuovi algoritmi ed architetture* di machine learning per robot che apprendono cumulativamente. In particolare, il progetto ha l'obiettivo di ottenere un progresso sostanziale in tre aree distinte ma legate rappresentate dai tre principi dell'ipotesi centrale del progetto: (a) architetture gerarchiche; (b) motivazioni intrinseche basate sulla rilevazione di novità ; (c) astrazione percettiva ed attenzione.

*Integrare le conoscenze ottenute* con gli esperienti empirici, i modelli bio-vincolati, ed i modelli di machine learning, ed applicare queste conoscenze alla costruzione di robot reali che dimostrano di possedere abilità di apprendimento cumulativo. Questo comporterà l'uso della piattaforma robotica iCub per lo sviluppo di due demo: CLEVER-B, una demo con orientamento tecnologico relativo ad un robot che agisce in un ambiente domestico (manipolazione di utensili da cucina), e CLEVER-B, una demo che mira a riprodurre e spiegare i risultat degli esperimenti condotti con le scimmie ed i bambini.

### *Riconoscimenti*

IM-CLeVeR è supportato dalla Commissione Europea nell'ambito dell'iniziativa “FP7 - Cognitive Systems, Interaction, and Robotics”, con il contratto numero 231722.

# *L'influenza delle perturbazioni sul paesaggio degli attrattori di una rete booleana casuale*

**Alessia Barbieri**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia
alessia.barbieri@gmail.com

**Marco Villani**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia.
*European Centre for Living Technology,*
S.Marco 2940 – 30124 Venezia.
marco.villani@unimore.it

**Roberto Serra**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative,*
*Università di Modena e Reggio Emilia;*
via Allegri, 9 Reggio Emilia.
*European Centre for Living Technology,*
S.Marco 2940 – 30124 Venezia.
roberto.serra@unimore.it

## *1. Introduzione*

Le reti booleane casuali, proposte più di 40 anni fa da Stuart Kauffman, rappresentano uno dei modelli più noti di sistemi complessi [1]. Esse si sono rivelate particolarmente utili per descrivere diverse importanti proprietà delle reti di regolazione genica.

Una ipotesi di grande generalità è quella secondo cui i sistemi soggetti a pressione evolutiva tendono a vivere in condizioni al confine tra ordine e disordine in uno stato che consente l'evoluzione senza mettere a rischio le loro caratteristiche vitali. Questi stati vengono chiamati critici. Fra le diverse nozioni di criticità che si trovano in letteratura, particolarmente interessanti sono quelle legate alla dinamica. In questi casi è

possibile dare una definizione precisa del concetto: esistono regioni dello spazio dei parametri di un sistema dinamico nelle quali è possibile osservare un comportamento ordinato del sistema ed esistono regioni in cui è possibile osservare comportamenti disordinati; gli stati dinamicamente critici sono quelli che corrispondono a regioni intermedie dello spazio dei parametri, tra ordine e caos. I sistemi in stati critici presentano importanti vantaggi sia rispetto ai sistemi caotici, i quali sonoipersensibili a piccole perturbazioni, sia rispetto a sistemi molto ordinati, i quali sono incapaci di adattarsi ai mutamenti dell'ambiente. I sistemi critici, invece, permettono alle perturbazioni di apportare piccole modifiche, senza tuttavia collassare.

Diversi studi evidenziano che, in funzione del valore di un preciso parametro, la dinamica tipica delle RBN manifesta una transizione fra ordine e disordine, ed è stato possibile individuare con precisione la regione critica. Recentemente, è stato inoltre possibile studiare alcuni sistemi biologici, dimostrando che le RBNs critiche (o ordinate ma molto vicino allo stato critico) sono in grado di descrivere alcuni importanti dati sperimentali, in particolare la distribuzione delle perturbazioni nei livelli di espressione genica del lievito S. cerevisiae in seguito a operazioni di knock-out di singoli geni [2][3][4] e l'andamento temporale delle attivazioni dei geni della linea cellulare HeLa [5].

Il lavoro è organizzato come segue: nella sezione 2 viene richiamato brevemente il modello delle reti booleane casuali, nella sezione 3 vengono introdotte le reti booleane casuali soggette a rumoreenella sezione 4 vengono riportati i principali risultati, commentati nella sezione 5.

## *2. Reti booleane e attrattori*

Le reti booleane casuali (RBN) sono state descritte in molti lavori (ad esempio [1]), qui riassumiamoquindi solo le scelte compiute per i nostri studi. La rete è composta da N nodi, ognuno dei quali può assumere valore 0 o 1. Lo stato del sistema è definito dal vettore booleano $X$ le cui componenti$X_i$ indicano lo stato del nodo $i$-esimo.

Si usa qui il cosiddetto modello *quenched*, in cui sono costanti sia le connessioni fra nodi che le funzioni booleane che determinano lo stato del nodo all'istante $t+1$ in funzione di quelli dei suoi ingressi all'istante precedente. Si tratta quindi di un sistema deterministico a stati finiti (se N è finito) i cui stati asintotici sono cicli (o punti fissi, ossia cicli di lunghezza 1).

Per una specifica rete, ogni nodo ha esattamente $k$ connessioni in ingresso, provenienti da $k$ nodi diversi scelti casualmente con probabilità uniforme tra i rimanenti N-1 nodi. Le funzioni booleane sono generate assegnando casualmente, ad ogni combinazione di valori di ingresso, valore 1 con probabilità b (e valore 0 con probabilità 1-b).

Studiando i comportamenti tipici di popolazioni di reti, sono stati individuati 3 regimi dinamici: *i)* reti ordinate, in cui la lunghezza dei cicli cresce lentamente col numero dei nodi, *ii)* reti caotiche, in cui la crescita è di tipo esponenziale e *iii)* reti critiche, che costituiscono l'elemento di separazione delle due classi. Data l'ipotesi che le reti critiche possano avere notevoli vantaggi rispetto alle altre, il nostro studio è in massima parte concentrato su reti critiche.

Una particolare importanza è stata attribuita fin dall'inizio agli attrattori della rete, che Kauffman identifica coi diversi tipi cellulari negli organismi pluricellulari. Nel caso di organismi unicellulari gli attrattori possono invece corrispondere a diverse possibilità di funzionamento coerente della cellula.

L'analogia deriva, in primo luogo, dall'osservazione che ogni cellula del nostro organismo ha lo stesso genoma e si differenzia solo per il fatto cheesprime gruppi di geni differenti, così come tutti gli attrattorisono composti dallo stesso insieme di elementi e si differenzianosolo per il fatto che nei diversi cicli sono attivati o disattivati elementi diversi.

### *3. Reti booleane rumorose*

Recentemente è stato fatto osservare [6] che gli attrattori di una rete booleana deterministica possano non essere adeguati per descrivere sistemi soggetti a rumore, come accade in effetti nelle reti di regolazione genica. Ad esempio, in alcuni casi il numero di molecole di un enzima, in virtù di fluttuazioni casuali, può diventare tanto piccolo da risultare inefficace. Risulta quindi interessante studiare il comportamento di reti booleane soggette a rumore, nel caso particolare in cui il rumore consiste in una perturbazione transitoria che dura un solo *timestep*ed influenza un solo nodo della rete (*single flip*).

Se si considerano quindi come sostanzialmente indistinguibili due attrattori tali che possano essere raggiunti con singoliflip [6], si è portati ad introdurre la nozione di insieme rilevante, presentata e discussa in [7].

Nel caso del singolo flip, la grande maggioranza delle reti presenta un solo insieme rilevante. Verrebbe quindi meno la possibilità di identi-

ficare gli insiemi rilevanti coi diversi tipi cellulari corrispondenti ad un unico genoma. Tuttavia è possibile evitare questa conseguenza se si suppone che i flip siano relativamente rari, per cui se da un attrattore A si può arrivare ad un attrattore B solo attraverso il flip di un singolo specifico gene allora è possibile ignorare questa eventualità, poiché la probabilità di osservare quello specifico flip nel corso della vita di una cellula è trascurabile. Solo quegli attrattori che possono essere raggiunti da un certo numero di flip sono effettivamente connessi. In questo modo, variando il valore di soglia sul numero minimo di flip che vengono ritenuti sufficienti a stabilire una connessione fra gli attrattori, è possibile individuare diversi insiemi rilevanti - quindi diversi tipi cellulari [7].

Questo per quanto riguarda le perturbazioni single flip. Studi preliminari indicano che il sistema tende a comportarsi in maniera qualitativamente analoga quando vengono permessi due flip simultanei. Un'altra variante importante riguarda invece perturbazioni di durata finita, e lo studio del comportamento degli insiemi di attrattori in queste circostanze è l'oggetto del presente lavoro.

Le perturbazioni che considereremo sono tali per cui lo stato di attivazione di un gene viene imposto dall'esterno per un certo periodo di tempo. Noi supponiamo che la durata della perturbazione (che può rappresentare ad esempio la presenza di una sostanza chimica o di uno stato ambientale particolare) sia tale da consentire alla rete perturbata di raggiungere un nuovo statostazionario. In seguito la perturbazione viene rilasciata, e si studia verso quale attrattore della rete originale i sistema rilassa: in quanti casi torna all'attrattore di partenza, e in quanti invece evolve verso un diverso ciclo?

Lo studio prevede dunque le seguenti fasi. Inizialmente, a partire da una condizione iniziale arbitraria, si lascia evolvere la rete imperturbata finché essa raggiunge l'attrattore nel cui bacino di attrazione ricade la condizione iniziale utilizzata. Indichiamo con A questo attrattore. A quel punto si sceglie a caso un nodo (sia esso l'*i*-esimo) e si fissa il suo valore a 0 se il suo valore $x(i) = 1$, a 1 altrimenti.

Si lascia poi evolvere la rete, in presenza della perturbazione costante, finché essa non raggiunge un attrattore, che è diverso da tutti quelli della rete imperturbata. Infatti la rete perturbata, con un nodo che ha valore fissato, è equivalente ad una rete in cui la funzione booleana di quel nodo assume costantemente valore 0 o 1.

Noi definiamo un attrattore della nuova rete perturbata A' *equivalente* al vecchio attrattore A se A e A' coincidono (previa una opportuna scelta della fase di eventuali cicli) in tutti i nodi tranne l' *i*-esimo.

In seguito la perturbazione viene rimossa, e la rete tende necessariamente ad uno degli attrattori della rete imperturbata: esso può coincidere con A oppure essere un diverso attrattore B.

## *4. Risultati*

Lo studio ha coinvolto l'analisi di reti critiche (con $k=2$ e $b=0,5$) sia di piccole dimensioni, con numero di nodi N=10, le quali consentono un'analisi esaustiva degli attrattori, sia di reti con N=100 per le quali si è ricorso ad un campionamento casuale dell'insieme delle possibili condizioni iniziali per determinare gli attrattori (che sono quindi un sottoinsieme del totale). Nel caso N=10 si sono perturbati tutti i nodi, mentre nel caso N=100 si è perturbato il 20% di stati diversi di ogni attrattore identificato.

I principali risultati sono riassunti nella Figura 1, nella quale i tre grafici in prima riga riportano i risultati ottenuti da reti con N=10, mentre i tre grafici in seconda riga riportano i risultati ottenuti con reti con N=100. Per ognuna delle due dimensioni di rete analizzate abbiamo ottenuto i seguenti grafici:

- il diagramma della "sensibilità" media della rete rispetto alla perturbazione riporta, in percentuale, quante volte la rete perturbata (inizialmente nel ciclo A) cambia stato asintotico (si porta in uno stato A' non equivalente ad A) e quante volte invece non è sensibile alla perturbazione e rimane nello stesso ciclo.
- Il diagramma centrale "Vecchi – nuovi attrattori"si riferisce ai soli casi in cui la rete è risultata sensibile alla perturbazione, ossia ai casi in cui A' non è equivalente ad A, ed indica quante volte l'attrattore A' è equivalente ad uno degli altri attrattori della rete imperturbata (caso indicato come "vecchio attrattore") e quante volte si tratta di un attrattore nuovo.
- Il terzo diagramma, "Ritorno all'attrattore di partenza",si riferisce solo ai "nuovi attrattori" del secondo diagramma e descrive ciò che avviene dopo che la perturbazione è stata rilasciata ed il sistema è tornato ad un attrattore della rete imperturbata, sia esso B. Il diagramma indica quante volte B coincide con A (caso indicato con sì) e quante volte inveceB≠A (no). Si noti che in un numero limitato di casi con N=100 non èstatopossibile individuare l'attrattore nei limiti di durata delle simulazioni imposti per esigenze computazionali**.**

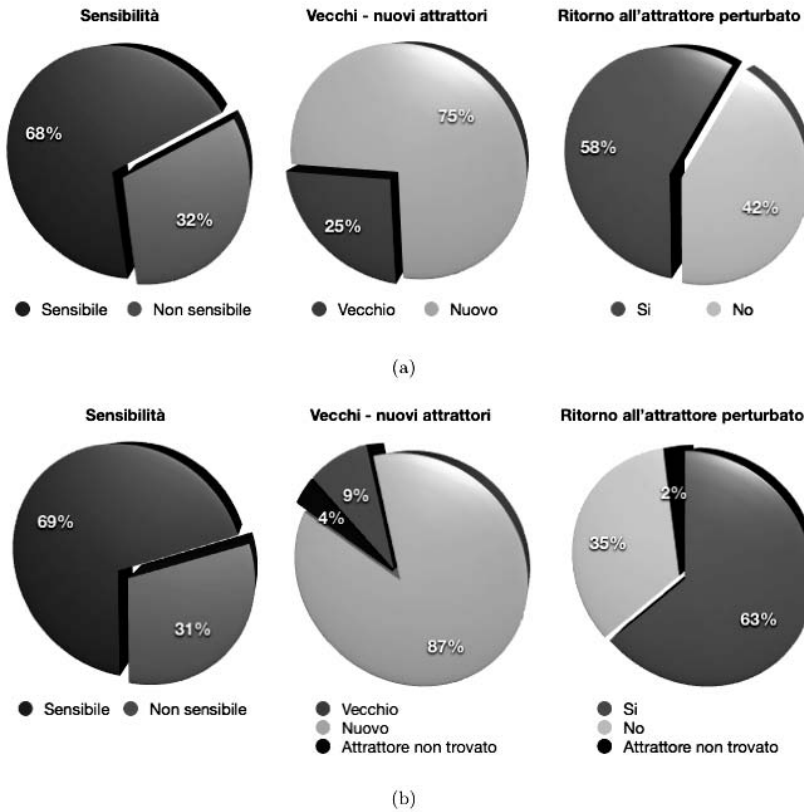



Figura 1: Comportamento medio di una RBN soggetta a perturbazione semi-permanente: i diagrammi della sezione (a) mostrano i risultati ottenuti analizzando reti con N=10; i diagrammi della sezione (b) mostrano gli stessi risultati ottenuti da reti con N=100.

## *5. Commenti*

Le principali considerazioni che si possono trarre dai risultati delle simulazioni sono le seguenti:

1. la sensibilità (come sopra definita) appare indifferente alle dimensioni della rete, nel range considerato;

2. al contrario, la tendenza a cadere in un attrattore della rete perturbata A' equivalente ad uno degli attrattori di partenza della rete imperturbata (diverso da quello iniziale A) cala al crescere delle dimensioni;

3. è interessante osservare come le perturbazioni abbiano un effetto rilevante: anche dopo che esse sono state rimosse la rete si porta in un attrattore diverso da quello iniziale quasi nel 30% dei casi per la rete da 100 nodi e quasi il 40% delle volte nel caso della rete da 10 nodi. Questo sembra indicare (sebbene con notevoli incertezze legate alla dimensione finita del campione esaminato) una maggior resilienza delle reti con dimensioni maggiori.

Questi risultati suggeriscono interessanti possibilità di verifica sperimentale in ambito biologico, mentre dal punto di vista teorico in futuro si potranno eseguire per le perturbazioni analisi analoghe a quelle che sono state compiute nel caso dei flip singoli, definendo le nozioni di insieme rilevante di attrattori e descrivendone le proprietà di rete.

## *Bibliografia*

1. Kauffman, S.A.: Origins of order. Oxford University press, Oxford, 1993
2. Serra R., Villani M.,Semeria, A.: Genetic network models and statistical properties of gene expression data in knock-out experiments. J. Theor. Biol 247, 149-157 (2004)
3. Serra R., Villani M., Graudenzi, A., Kauffman, S.: Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. J. Theor. Biol. 246, 449-460 (2007)
4. Ramo P. Kesseli J., Yli-Harja O.: Perturbation avalanches and criticality in gene regulatory networks. J. Theor. Biol. 242, 164-170 (2006)
5. Schmulevich, A., Kauffman, S.A., Aldana, M.: Eukaryotic cells are dynamically ordered or critical but not chaotic. PNAS 102, 13439-13444 (2005)
6. Ribeiro, A.S., Kauffman, S.A.: Noisy atractors and ergodic sets in models of gene regulatory networks. J. Theor. Biol. 247, 743-755 (2007)
7. Barbieri A., Villani M., Serra R., Kauffman S.A., Colacci A.: The influence of noise on the dynamic of random boolean networks. In Morabito, C. (ed): Proceedings of Wirn 2009. Amsterdam: IOS Press (2009, in press)

# *Ribocell Modeling*

**Pierluigi Della Gatta**
*Dipartimento di Chimica, Università di Bari*;
Via Orabona, 4
70125 Bari, Italy
Tel. +39 0805442054, Fax +390805442129
pierluigi.dellagatta@yahoo.com

**Fabio Mavelli** (corresponding author)
*Dipartimento di Chimica, Università di Bari*;
Via Orabona, 4
70125 Bari, Italy
Tel. +39 0805442054, Fax +390805442129
mavelli@chimica.uniba.it

## *1. Introduzione*

La biologia sintetica è una disciplina emergente [1,2] che ha frai suoi principali obbiettivi sintesi di strutture cellulari semplificate che possono essere considerate viventi secondo una definizione minimale di organismo vivente (*minimal cell*) [3]. Affinché una struttura supramolecolare possa essere considerata vivente essa deve perpetuarsi in un regime stazionario di continua sintesi e ricambio dei propri costituenti (*self-mantainance*) ed eventualmente crescere e riprodursi (*self-reproduction*) come effetto del proprio metabolismo. La possibilità di un comportamento evolutivo (*evolvability*) è invece una prerogativa più di una popolazione di organismi che di un singolo individuo e può essere espletata solo attraverso un meccanismo di selezione darwiniana mediante trasmissione di informazione dalla cellula madre alla figlia con una certa percentuale di errore.

In questo contesto è stato recentemente introdotto un modello teorico di cellula minima concettualmente molto semplice che è la ribocellula[4]. Essa è costituita da una vescicola lipidica al cui interno sono presenti due ipotetici ribozimi: uno (RNA- Polimerasi)in grado di catalizzare la polimerizzazione dei ribozimi stessi partendo da un filamento di RNA che funge da stampo e dai nucleotidi, l'altro(RNA-Sintasi), in-

vece, responsabile della sintesi dei lipidi di membrana a partire da molecole precursori (Figura 1). In un ambiente di reazione ricco di nucleotidi e precursori lipidici, la ribocellula potrebbe mostrare proprietà di auto-mantenimento, auto-riproduzione ed evoluzione.

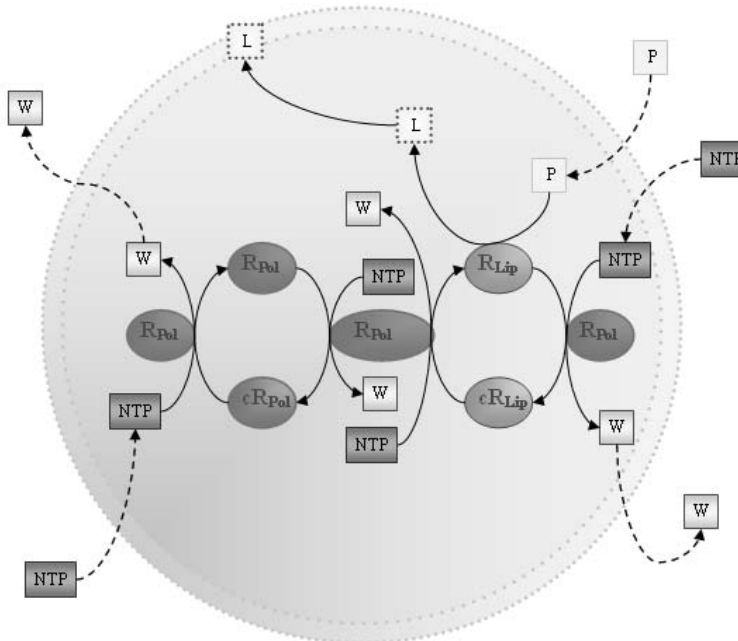



Figura 1 - Rappresentazione grafica del metabolismo della ribocellula: i processi di trasporto attraverso la membrana sono rappresentati da linee tratteggiate, le reazioni da linee solide.

Sebbene quindi la Ribocellula sia al momento soltanto un modello ipotetico di cellula vivente, lo studio in silico di questo sistema è volto alla definizione dei range di efficienza che l'RNA-Polimerasi e l'RNA-Sintasi devono esibire per poter osservare l'istaurarsi di un regime di auto-mantenimento e auto-replicazione quando accoppiati con la dinamica delle membrane lipidiche. In particolare verrà investigata la possibilità che si instauri una sincronizzazione fra i processi di duplicazione della membrana e del genoma minimo della Ribocella.

## *2. Il modello cinetico*

Il modello cinetico utilizzato per descrivere il comportamento dinamico di una vescicola lipidica è in accordo con un modello recentemente proposto per vescicole reagenti [5] che è stato applicato in simulazione di processi competitivi [6,7] di vescicole acido oleico e POPC [6, 9]. In questo lavoro, però, l'evoluzione temporale è descritta con un approccio deterministico, anzicchè stocastico, assumendo che tutte le vescicole abbiano in media lo stesso comportamento temporale.

### *2.1 La vescicola lipidica*

La vescicola viene trattata come un reattore a volume variabile $V_C$ che, attraverso la sua membrana, scambia acqua con l'esterno sotto un gradiente di pressione osmotica. La superficie della membrana $S_\mu$ è, invece, in equilibrio con le molecole di lipide presenti nell'ambiente acquoso grazie a rapidi processi di associazione e rilascio. In presenza della sintesi di nuove molecole di lipide queste vengono rapidamente assorbite per ripristinare la condizione di equilibrio facendo aumentare la superficie della bilayer. Lo stato della membrana può essere, quindi, monitorato attraverso la superficie ridotta: $\Phi = S_\mu / (36\pi V_C^2)^{1/3}$. Per una vescicola perfettamente sferica risulterà $\Phi = 1$, mentre $\Phi < 1$ se l'aggregato lipidico risulta rigonfiato ed in uno stato ditensione elastica che può portarlo a scoppiare (crisi osmotica). Tale fenomeno può capitare al di sotto di un certo valore di tolleranza: $\Phi < 1 - \varepsilon$, sperimentalmente determinabile per differenti tipi di lipide (as esempio per l'acido oleico $\varepsilon = 0.21$ [6, 9]). Se $\Phi > 1$ la vescicola è ripiegata e può andare incontro a un processo di divisione. In questo lavoro assumeremo che una divisione spontanea avviene ogni qual volta sia possibile formare due vescicole figlie entrambe sferiche, ossia se $\Phi = \sqrt[3]{2}$.

Attraverso la membrana sono anche possibili processi di trasporto di substrati guidati dal gradiente di concentrazione. Per semplicità si è assunto che la variazione di concentrazione avvenga solo a cavallo della membrana lipidica e che sia il volume acquoso interno che l'ambiente esterno siano omogenei.

### *2.2 Il metabolismo interno*

Il meccanismo cinetico proposto per la Ribocellula è riportato in Figura 2. Entrambe le coppie di filamenti di RNA danno vita ad una

reazione di associazione reversibile (1) che porta alla formazione di un dimero. Tale equilibrio, dipendente fortemente dalla temperatura, è fortemente spostato dalla parte della forma associata.

$$\begin{aligned} T + T' &\underset{k_{ST}}{\overset{k_{DT}}{\rightleftharpoons}} TT' \\ R + R' &\underset{k_{SR}}{\overset{k_{DR}}{\rightleftharpoons}} RR' \end{aligned} \qquad (1)$$

$$\left.\begin{aligned} R + S &\xrightarrow{k_{RS}} R@S \\ R@SS'_n + B_{n+1} &\xrightarrow{k_{B_{n+1}}} R@SS'_{n+1} + W \\ R@S_N S'_N &\xrightarrow{k_{dis}} R + SS' \end{aligned}\right\}_{S=R,R',T,T'} \qquad (2)$$

$$P + T \longrightarrow T + L + W \qquad (3)$$

Figura 2: Schema cinetico del metabolismo interno della ribocellula: (1) associazione reversibile dei filamneti di RNA-Polimerase (R) e RNA-Sintase (T) con i rispettivi filamenti complementari R' e T'; (2) ciclo catalitico della replicazione degli strand di RNA; (3) conversione del precursore P nel Lipide di membrana L catalizzata dal ribozima T.

La replicazione degli strand di RNA, è catalizzata dal ribozima Polimearsi R secondo gli stadi (2), in parentesi. Il ciclo inizia con R che si lega a un stampo S, disponibile in forma monomerica, portando alla formazione del complesso R@S. Tale complesso darà inizio alla polimerizzazione del filamento coniugato S', mediane accopiamente delle basi complementari e liberando il sottoprodotto W. Quando lo strand S' è stato duplicato completamente, il ribozima Polimerasi libera il dimero. Per poter ottenere duplicare il "genoma" il ciclo catalitico dovrà operare su tutti e 4 i filamenti di RNA almeno una volta. Va inoltre sottolineato come, affinché ciò sia possibile devono essere presenti contemporaneamente almeno 3 filamenti: necessariamente R, e un filamento T o T' e un secondo R o R'.

In fine, la reazione (3) è la conversione del precursore P nel lipide L catalizzata dal ribozima T.

*2.3 Il vettore di stato e il sistema ODE*

Lo stato del Ribocella è descritto dal vettore:

$$\mathbf{x}^T(t) = \left(n_1^C, n_2^C, \ldots n_N^C, n_L^\mu, V_C\right)$$

dove $n_i^C$ e $n_L^\mu$ sono i numeri medi di molecole delle specie $X_\mathrm{i}$ ($i$=1,2 …$N$)contenute nel core acquoso e di lipide nella membrana, mentre $V_\mathrm{C}$ rappresenta il volume acquoso. La superficie della membrana è data da $S_\mu = \alpha_L n_L^\mu / 2$, dove $\alpha_\mathrm{L}$. rappresenta l'area della testa del lipide e 1/2 tiene conto del doppio strato della membrana. Il sistema di equazioni differenziali da risolvere è quindi dato da:

$$\begin{cases} \dfrac{dn_i^C}{dt} = N_A V_C \displaystyle\sum_{\rho=1}^{R}\left(b_{i,\rho} - a_{i,\rho}\right) r_\rho + P_i S_\mu \left( \left[X_i\right]^{Ex} - \dfrac{n_i^C}{N_A V_C} \right) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, 2 \ldots N \\ \dfrac{dn_L^\mu}{dt} = k_{in} S_\mu \dfrac{n_L^C}{N_A V_C} - k_{out} n_L^\mu \\ \dfrac{dV_C}{dt} = P_{water} S_\mu \left(C^C - C^{Ex}\right) \end{cases}$$

dove $N_A$ numero di Avogadro, $R$ numero di reazione del metabolismo interno, **b** e **a** matrici dei coefficienti stechiometrici [10], $P_\mathrm{i}$ e $P_\mathrm{W}$ permeabilità rispettivamente della specie $X_\mathrm{i}$ e dell'acqua, $[X_\mathrm{i}]^{Ex}$ concentrazione acquosa esterna, $r_\rho$ velocità di reazione dello stadio metabolico ρ-esimo:

$$r_\rho = k_\rho \prod_{j=1}^{N} \left( n_j^C \,/\, \left(N_A V_C\right)\right)^{a_{j,\rho}}$$

con $k_\rho$ costante cinetica di reazione. $C^\mathrm{C}$ e $C^\mathrm{Ex}$ rappresentano la concentrazione totale interna ed esterna di tutte le specie. Le concentrazioni esterne $[X_\mathrm{i}]^{Ex}$ sono assunte costanti nel tempo per tutte le specie, così come $C^{Ex}$. Poiché tutte le specie intermedie R@SS'$_\mathrm{n}$ vengono considerate esplicitamente, il numero di equazioni differenziali nel sistema è strettamente dipendente dalla lunghezza dei ribozimi utilizzati. E' stato, quindi, realizzato un programma MATLAB: RIBOCEL, per la costruzione del sistema ODE e per la sua risoluzione numerica sulla base della sequenza dei ribozimi R e T ipotizzati. Questo programma è stato u-

tilizzato per risolvere numericamente il sistema ODE mediante tutti gli algoritmi numerici implementati in MATLAB imponendo i seguenti vincoli:

$$n_i^C \geq 0 \qquad (i = 1, 2, ...N)$$
$$1 - \varepsilon \leq \Phi \leq \sqrt[3]{2}$$

Nel caso in cui si verifichi che $\Phi > \sqrt[3]{2}$ la vescicola subisce una divisione in due figlie identiche e tutte le variabili di stato del sistema vengono, quindi, dimezzate. Negli altri casi il calcolo viene fermato, o perché è avvenuta una crisi osmotica $\Phi < 1 - \varepsilon$, o perché si è raggiunto uno stato fisicamente non significativo.

Tabella 1: Costanti Cinetiche e Permeabilità della Ribocellula

| Constanti Cinetiche | Valori | Descrizione | Rif. |
|---|---|---|---|
| $k_{SS}$·[$s^{-1}M^{-1}$] | $8.8 \cdot 10^{6}$ | Formazione RR' e TT' | [11] |
| $k_S$[$s^{-1}$] | $2.2 \cdot 10^{-6}$ | Dissociazione RR' e TT' | [11] |
| $k_{R@S}$[$s^{-1}M^{-1}$] | $5.32 \cdot 10^{5}$ | Formazione R@R/R' e R@T/T' | [14] |
| $k_{R@SS}$[$s^{-1}$] | $9.9 \cdot 10^{-3}$ | Dissociazione Complessi R@RR' e R@TT' | [14] |
| $k_B$[$s^{-1}M^{-1}$] | 0,113 | Polimerizzazione Nucleotide | [14] |
| $k_L$ [$s^{-1}M^{-1}$] | 0,017 | Costante di Formazione Lipide | [12] |
| $k_{in}$ [$dm^2s^{-1}$] | $7.6 \cdot 10^{19}$ | Costante di associazione del lipide alla membrana | [9] |
| $k_{out}$ [$dm^2s^{-1}$] | $7.6 \cdot 10^{-2}$ | Costante di rilascio del lipide dalla membrana | [9] |
| Permeabilità [cm·$s^{-1}$]x$10^{-8}$ | Valori | Descrizione | |
| $P_P$ | 0.42 | Permeabilità Precursore Lipide | |
| $P_A$=$P_U$=$P_C$=$P_G$ | 0.0019 | Permeabilità Nucleotidi | [14] |
| $P_W$=$P_T$=$P_R$ | 0.0 | Permebilità W, T, T', R e R' | |
| $P_{water}$ | $1.0 \cdot 10^{6}$ | Permeabilità acqua | [13] |

*2.4 Costanti Cinetiche e permeabilità*

In Tabella 1 sono mostrati i valori di tutte le costanti cinetiche e delle permeabilità utilizzati in questo lavoro insieme con i riferimenti da cui sono stati derivati.

I due ribozimi sono assunti entrambi lunghi 20 nucleotidi (numero di basi minimo per una conformazione ripiegata) con una sequenza di basi casuale, e con un comportamento cinetico simile. Le costanti cinetiche di formazione $k_{SS}$ e dissociazione $k_S$ di entrambi i dimeri (S=R e T) sono state poste uguali ai valori determinati per una sequenza di 10 nucleotidi [11] .

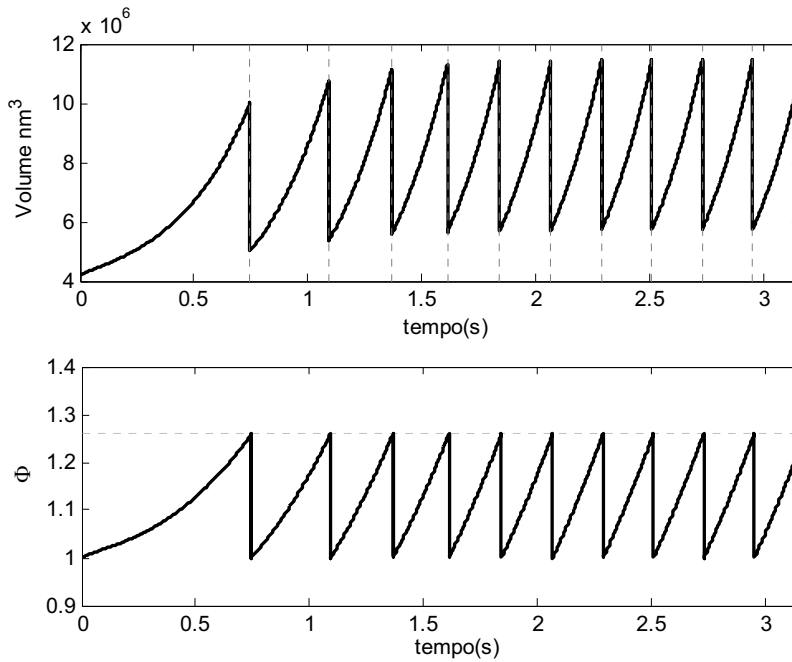



Figura 3: Evoluzione temporale della Ribocella nel primo anno di vita. Sono riportati: in alto il volume interno (le linee verticali rappresentano i tempi di divisione), in basso la superficie ridotta (la linea orizzontale rappresenta la condizione di divisione $\Phi = 2^{1/3}$ ).

Le costanti cinetiche sia per la formazione dei complessi R@S che di dissociazione dei complessi R@SS' (S=R, R', T e T') sono state poste uguali a quelle misurate per l'enzima umano β Polimerasi [14] rispettivamente per l'associazione con un singolo strand di DNA e per la dissociazione da un dimero.

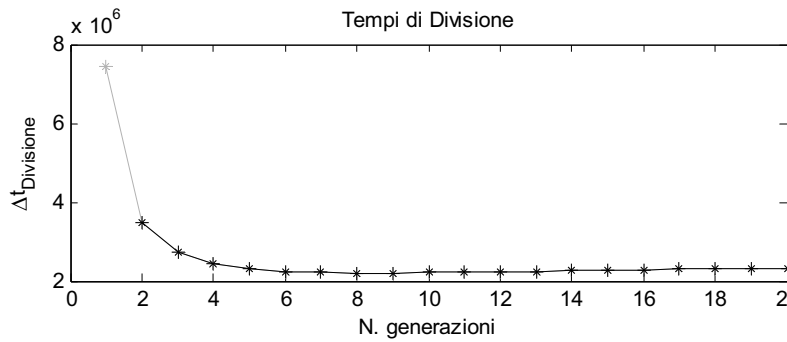



Figura 4: Evoluzione temporale della Ribocella: andamento dei tempi di divisione contro il numero di generazioni. Il tempo di divisione si attesta su 26.6 giorni

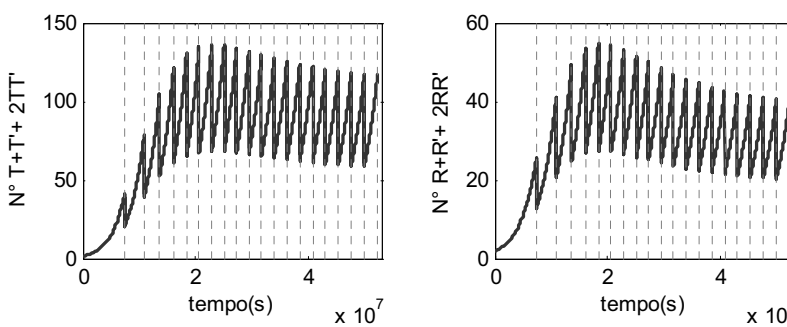



Figura 5: Evoluzione temporale della Ribocella: andamento della popolazione dei Ribozimi nel tempo, le linee trattegiate verticali indicano i tempi di divisione.

La costante cinetica per la sintesi catalitica del lipide è stata assunta unguale a quella della reazione di *splicing*, catalizzata dal ribozima *Hammerhead* [12].

Il comportamento cinetico dei differenti Nucleotidi è stato considerato lo stesso (pur manenendoli differenziati) e a $k_L$ è stato assegnato il valore ottentuto da dati sperimentali [14]. Dallo stesso lavoro è stata anche recavata la permeabilità dei nucleotidi, mentre la membrana è stata assunta impermeabile al materiale genetico. Le costanti cinetiche di scambio del lipide sono prese da un lavoro precedente [9].

Gli unici parametri cinetici assegati arbitrariamente sono la permeabilità del sottoprodotto W posta uguale a zero (compatibile con l'ipotesi di una specie carica) e quella del precursore, il cui valore però, è paragonabile a quello riportato per diversi composti organici (p.e: l'arabitolo) nel caso di membrane di acido oleico [13].

## *3. Risultati*

I calcoli sono stati effettuati per una vescicola di acido oleico (area della testa $\alpha_L$=0.3nm$^2$; tolleranza osmotica $\varepsilon$=0.21, spessore membrana 4.0nm), inizialemente sferica $\left(\Phi = 1\right)$ con raggio 50nm, in cui si è supposto che al tempo zero fossero presenti solo i due dimeri RR' e TT': $[RR']_0^C = [TT']_0^C = 3.6\mu M$. Le concentrazioni interne ed esterne di tutti i nucleotidi sono state poste uguali al valore 500μM (in analogia con un esperimento di duplicazione autocatalitica del DNA [14]) come anche quella del precursore *P*, mentre la concentrazione del sottoprodotto della condensazione *W* è stata posta uguale a zero e quella del lipide uguale al valore di equilibrio $[L]_0^C = [L]_0^{Ex} = 66.7\mu M$ [9]. 0.3M è la di osmoliti inerti. Tutte le concentrazioni esterne sono state assunte costanti grazie ad un continuo flusso di substrati dall'esterno. Al tempo zero, quindi, la vescicola è stata posta in una condizione di perfetto bilanciamento osmotico ($C^C$=$C^{Ex}$).

La Figura 3 mostra l'evoluzione temporale della Ribocella in una finestra di un anno mostrando come si instauri un regime ciclico di fasi di crescita e divisione cellulare. Dopo una decina di generazioni, questo regime raggiunge una condizione di stazionarietà in cui la ribocellula raddoppia il suo contenuto e le sue dimensioni durante la fase di crescita, ma genera sempre due cellule figlie sferiche di raggio pari a circa 110nm. Il grafico in Figura 4 riporta, invece l'andamento dell' intervallo

temporale fra due divisioni successive, in un arco di 20 generazioni, mostrando come esso si stabilizzi intorno al valore di 26.6 giorni.

Le popolazioni medie dei ribozimi sono riportate nell'arco delle prime 20 generazioni in Figura 5. Alla fine del ciclo di crescita il numero totale di entrambi Ribozimi R/R'e T/T' raddoppia e, al raggiungimento del regime di duplicazione stazionario, tale numero, dopo ogni divisione, rimane costantemente uguale rispettivamente a 20 e 59 unità mostrando un comportamento perfettamente sincronizzato con quello della membrana. La differenza osservata delle due popolazioni va ascritta alla minore disponibilità dei filamenti *R* a fungere da stampo essendo coinvolti come catalizzarori.

## *4. Conclusioni*

In questo lavoro è stato presentato un modello cinetico per la ribocellula, un prototipo ipotetico di cellula minima, al fine di studiarne il comportamento temporale sulla base di valori delle costanti cinetiche e delle permeabilità assegati da dati riportati in letteratura. Sebbene questo studio sia tuttora in corso e richieda un maggiore approfondimento della dipendenza del comportamento temporale della ribocellula da parametri cinetici e strutturali (lunghezza dei ribozimi, dimensione della vescicola, ecc.), pur tuttavia esso mostra, in via preliminare, che una sincronizzazione fra la duplicazione del genoma e la riproduzione della membrana è possibile, all'interno delle approssimazioni utilizzate. L'elevato valore del tempo di divisione ottenuto: 26.6 giorni, va essenzialmente ascritto al basso valore utilizzato per la costante di dissociazione $k_S$, relativa alla dissociazione di un dimero di RNA a temperatura ambiente [11]. Infatti, un aumento della temperatura di lavoro potrebbe aumentare di molto l'efficienza della riproduzione come mostrato recentemente dal processo di replicazione auto-catalizzata di filamenti di RNA[15] la cui concentrazione in soluzione aumenta di 25 volte ogni 5 ore ad una temperatura di 42°C.

## *5. Bibliografia*

1. Benner S. A., Sismour A. M., Nat. Rev. Genet. 6, 533 (2005).
2. Luisi P. L., Ferri F., Stano P., Naturwissenschaften 93, 1727 (2006); Luisi P.L., Chemistry & Biodiversity 4, 603 (2007).
3. Luisi P.L., Orig.Life Evol. Biosphere 28, 613–622, (1998).

4. Szostak JW, Bartel DP, Luisi PL, Nature 409, 387 (2001).
5. Mavelli F., Ruiz-Mirazo K., Phil.Trans.R.Soc.B. 362, 1789-1802 (2007);
6. Chen I.A., Roberts R.W., Szostak J.W., Science, 305, 1474-1476 (2004).
7. Cheng, Z.; Luisi, P.L,. J. Phys. Chem. B 107 , 10940-10945 (2003).
8. Mavelli F., Ruiz-Mirazo K., In: Sergey M. Bezrukov. "Noise and Fluctuations in Biological, Biophysical, and Biomedical Systems". Bellingham, Washington: SPIE (United States), 6602, 1B1-1B10 (2007).
9. Mavelli, F., Lerario, M., Ruiz-Mirazo, K., In Arabnia, H. R. et al. (eds.) BIOCOMP'08 Proceedings (Vol II), CSREA Press, 934-941 (2008).
10. Mavelli F., Piotto S.J., Mol. Struct. 771, 55-64 (2006).
11. Christensen U., Biosci Rep. 27:327–333 (2007).
12. Stage-Zimmermann T. K. , Uhlenbeck O.C., RNA 4, 875-889 (1998).
13. Sacerdote M. G., Szostak J. W., PNAS 102, 6004-6008 (2005).
14. Mansy S., Szostak J.W., Nature 454, 122-126 (2008).
15. Lincoln T. A., Joyce G.F., Science 323, 1229-1232 (2009).

# *Foraging e Dimensione del Gruppo. Un Modello Computazionale del Comportamento Sociale dei Mammiferi Carnivori*

**Marco Campenni**
*Istituto di Scienze e Tecnologie della Cognizione*;
Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: ++39 06 4993 2201, Fax: ++39 06 4459 5243 -
Dipartimento di ricerche filosofiche, Università Tor Vergata di Roma;
Via Columbia, 1, 00133 Roma (RM)
Tel.: ++39 06 7259 5120, Fax: ++39 06 7259 5051
marco.campenni@istc.cnr.it

**Federico Cecconi**
*Istituto di Scienze e Tecnologie della Cognizione*; Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: ++39 06 4993 2201, Fax: ++39 06 4459 5243
federico.cecconi@istc.cnr.it

**Abstract** In questo lavoro viene presentato un modello computazionale di un fenomeno sociale fondamentale nello studio del comportamento animale: il foraging. Lo scopo di questo lavoro è, da una parte, testare la validità del modello proposto confrontandolo con un altro modello esistente, quello del flocking; dall'altra, cercare di capire se il modello può fornire indicazioni utili nello studio delle dimensioni del gruppo in alcune specie di mammiferi sociali carnivori.

***Parole chiave: foraging, etologia artificiale, swarm intelligence***

## *1. Optimal Foraging Theory*

La Foraging Theory è una branca della behavioral ecology che studia il comportamento di foraging degli animali in risposta ad un ambiente (più o meno) complesso nel quale essi vivono. La Optimal Fora-

ging Theory (OFT) (vedi [4][5]) è una versione raffinata di questa teoria che prevede che gli animali debbano trovare, catturare e consumare il cibo più calorico, impiegando il minor tempo possibile.

Essenzialmente esistono tre principali versioni della OFT:

- l'optimal diet model che descrive il comportamento di un forager che si imbatte in differenti tipi di prede e deve decidere quale attaccare;
- la patch selection theory che descrive il comportamento di un forager la cui preda è concentrata in piccole aree lontane l'una dall'altra in maniera che il forager deve spendere una quantità significativa di tempo (e di energia) per spostarsi dall'una all'altra;
- la central place foraging theory che descrive il comportamento di un forager che deve ritornare in un particolare posto (tipicamente sempre lo stesso) per consumare il proprio cibo o per farlo consumare alla propria prole.

Il settore di ricerca che si occupa di sviluppare modelli di comportamenti legati al foraging e all'hunting (la caccia) ha una lunga e consolidata tradizione ed opera da più di trent'anni (si vedano, ad esempio, [1][2][3][8][9]).

## 2. *Particle Swarm Optimization (PSO) Vs il nostro Modello*

### *2.1 PSO*

Il nostro modello, realizzato utilizzando NetLogo ([22]), è fortemente ispirato all'algoritmo di PSO.

L'algoritmo PSO è un algoritmo di ottimizzazione stocastico basato su popolazione che deriva dall'Evolutionary Optmization (EO). L'obbiettivo dell'EO è quello di determinare i valori per i parametri o per le variabili di stato di un modello che forniscano la migliore soluzione possibile ad una predefinita funzione costo o obbiettivo, o ad un set di funzioni, nel caso di due o più obbiettivi concorrenti ([12][10][6][20][13]).

Numerosi approcci sono stati proposti per trovare in maniera efficiente i) le migliori soluzioni a problemi con funzione mono-obbiettivo e ii) soluzioni Pareto-ottimali a problemi complessi di ottimizzazione multi-obbiettivo. In particolare, gli algoritmi evolutivi si sono dimostrati un potente approccio nel risolvere problemi di ricerca e ottimizzazione che considerassero obbiettivi multipli in conflitto fra loro.

Anche se la classe dei problemi di ottimizzazione multi-obbiettivo è stata investigata sufficientemente (vedi [19][15]), gli algoritmi evolutivi oggi disponibili tipicamente implementano un singolo algoritmo per l'evoluzione della popolazione. Tuttavia, le teorie esistenti (vedi [23][24]) e gli esperimenti numerici hanno dimostrato che è impossibile sviluppare un singolo algoritmo per l'evoluzione della popolazione che sia sempre efficiente per un set di diversi problemi di ottimizzazione. Così, negli ultimi anni sono stati proposti algoritmi memetici (anche chiamati algoritmi genetici ibridi) per aumentare l'efficienza degli algoritmi di ottimizzazione basati su popolazione. Questi metodi si ispirano ai modelli di adattamento nei sistemi naturali e utilizzano algoritmi genetici per l'esplorazione globale dello spazio della ricerca combinandoli con euristiche locali per lo sfruttamento delle aree più promettenti.

La versione originale del PSO è stata introdotta da Kennedy et al. (vedi [7][16]); nel PSO (vedi anche [17][18]) uno swarm (sciame) di particelle si muove nello spazio n-dimensionale delle soluzioni di un problema (multi-obbiettivo), cercando di trovare la soluzione migliore.

Questo algoritmo può essere utilizzato per modellare e prevedere il comportamento sociale in presenza di differenti obbiettivi. Lo swarm è generalmente modellato da particelle dotate di una posizione ed una velocità in uno spazio multidimensionale. Queste particelle "volano" in questo iperspazio ed hanno due principali capacità cognitive (molto semplici): i) hanno memoria della propria migliore posizione; ii) sono a conoscenza della posizione migliore in assoluto e di quella migiore rispetto alle particelle vicine. I membri dello swarm si comunicano l'uno con l'altro istantaneamente le posizioni ed aggiustano la propria posizione e velocità in conseguenza di tali informazioni.

In questo modo, una particella possiede le seguenti informazioni:

- un best globale che è noto ad ogni particella ed è aggiornato immediatamente ogni volta che una nuova posizione migliore viene trovata da una qualsiasi delle particelle nello swarm;
- un best locale che si riferisce alla migliore soluzione trovata dalla singola particella;
- un best dei vicini che la particella ottiene comunicando con un sottoinsieme dello swarm.

Per realizzare un modello di comportamento di foraging il tradizionale algoritmo Particle Swarm Optimization (PSO) presenta dei limiti che non ne permettono l'utilizzo. Il modello che presentiamo cerca di superare questi limiti e di cogliere gli elementi fondamentali che contribuiscono all'emergenza del comportamento sociale in questione, sotto cer-

te condizioni; per verificare la bontà del modello abbiamo confrontato i nostri risultati con quelli ottenuti con un tradizionale algoritmo di flocking.

### *2.2 Il Nostro Modello*

Anche nel nostro modello gli agenti (le particelle) hanno:

- una informazione locale (che deriva dalla loro percezione dell'ambiente) e
- una informazione (indiretta) che deriva dai vicini (prodotta dall'interazione con gli altri agenti);
- inoltre essi cambiano la loro posizione e velocità in base alle informazioni che ricevono.

Tuttavia, nel nostro modello non esiste alcuna informazione relativa ad un best globale: d'altro canto, assumendo che nel nostro mondo simulato non esista una differenza qualitativa fra una unità di cibo e l'altra, non ha senso parlare di best globale. Nel nostro modello esistono delle unità di cibo, qualitativamente uguali fra loro, che vengono distribuite nell'ambiente in modo tale da formare poche isole ricche di cibo (come nella OFT e più precisamente nella sua accezione di patch selection theory).

Inoltre abbiamo implementato una protoforma di comunicazione: gli agenti si scambiano dei segnali quando trovano una fonte di cibo e questi segnali si propagano nell'ambiente (degradandosi nell'intensità) come delle onde acustiche.

Un classico algoritmo di PSO non è utile a modellare il fenomeno di nostro interesse perchè:

1) nel comportamento di foraging in esame (*patch selection theory*) non ha senso parlare di best globale;
2) inoltre, in un tale comportamento sociale è poco plausibile parlare di informazione istantanea (come quella modellata dal PSO).

Per sopperire a questi due principali limiti del PSO, nel nostro modello non è presente alcuna informazione relativa al best globale e viene implementata una forma di comunicazione stigmergica fisicamente plausibile.

Il mondo è costituito da una griglia toroidale bidimensionale sulla quale si muovono gli agenti.

Il task degli agenti (robots simulati, vedi [11][14]) è quello di esplorare il mondo per trovare tutte le unità di cibo disponibili nel minor tempo possibile. Ciascun agente è caratterizzato da alcune proprietà individuali: una velocità v1 (di base); una velocità v2 (minore di v1, da adottare in una area dove è presente del cibo); una direzione; una velocità di rotazione (relativa a quanto bruscamente l'agente cambia direzione); un flag che indica che l'agente si trova in una area particolare (dove è presente del cibo).

Nel modello inoltre sono presenti delle semplici regole che contribuiscono a determinare il comportamento degli agenti:

i) se un agente A è vicino ad un agente B, entrambi cambiano la propria direzione in modo da allontanarsi il più possibile l'uno dall'altro;
ii) quando un agente trova del cibo, cambia la propria velocità (da v1 a v2) e
iii) cambia la propria direzione in modo da effettuare movimenti più circolari,
iv) emettendo un segnale che si propaga nell'ambiente come un'onda acustica.
v) Quando un agente riceve un segnale (che può essere il risultato della somma di segnali differenti), cerca di seguire il gradiente del segnale.

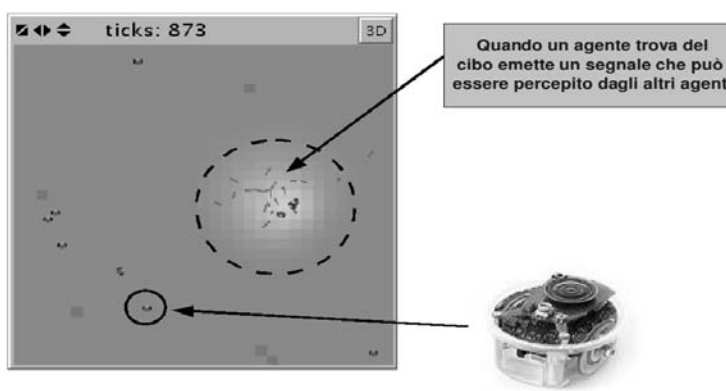



Figura 1. L'interfaccia grafica del simulatore. Gli agenti, i piccoli robots azzurri, si muovono nell'ambiente alla ricerca del cibo (quadratini marroni). Quando un agente trova una unità di cibo, emette un segnale che si propaga nell'ambiente (dentro il cerchio tratteggiato).

Nel nostro modello (vedi Figura 1) il segnale è differente dal segnale chimico emesso dalle termiti per segnalare agli altri membri della colonia la presenza di una fonte di cibo (vedi [21]); anche nel nostro modello il segnale è soggetto ad evaporazione (come nel caso delle termiti), ma (a) si propaga nell'embiente in modo concentrico rispetto all'origine dell'emissione e non viene depositato nell'ambiente sotto forma di traccia e (b) ciascun segnale può essere sommato ad altri segnali prodotti in altre parti del mondo; (c) inoltre il segnale viene emesso solo se l'agente modifica la propria velocità.

Una volta che il segnale viene emesso, gli agenti possono seguire il gradiente del segnale (ottenedo informazione indiretta sulla posizione della sorgente di cibo).

## *3. Risultati delle Simulazioni*

Per verificare la bontà dei risultati ottenuti utilizzando il nostro modello, abbiamo deciso di confrontarli con quelli ottenuti con un classico modello di flocking, già utilizzato per modellare comportamenti di foraging ([9]) e più plausibile di un tradizionale PSO.

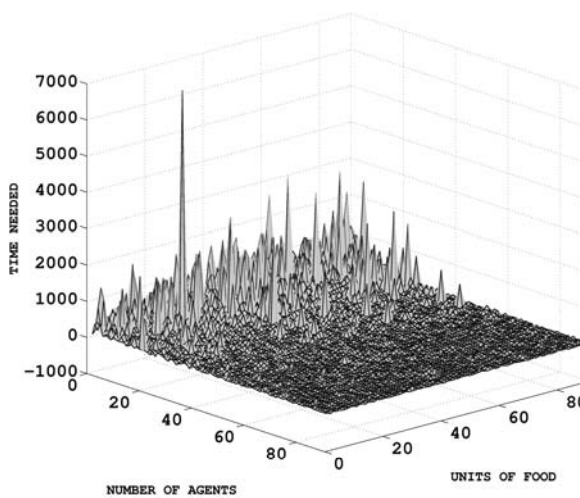



Figura 2a. Risultati del flocking

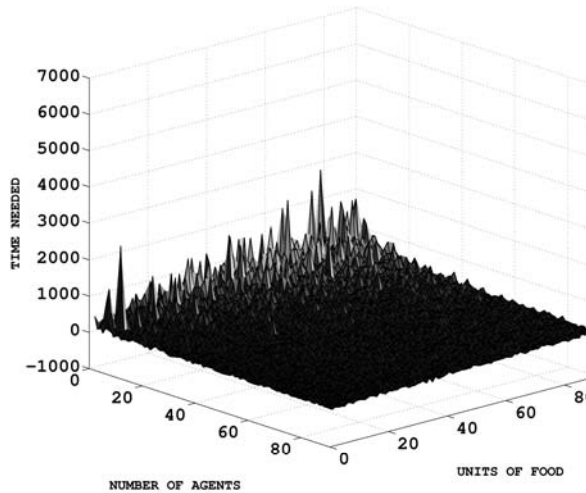



Figura 2b. Risultati del nostro modello

Figure 2a e 2b. Nelle figure 2a e 2b è rappresentata la superficie delle risposte (espresse in termini di tempo impiegato per trovare tutte le unità di cibo) nel caso del modello di flocking (2a) e in quello del nostro modello (2b) in cui abbiamo fatto variare il numero di agenti e di unità di cibo molto lentamente (da 10 a 100 con step di 1) utilizzando singoli run.

Come si può notare nelle Figure 2a e 2b il tempo impiegato per trovare tutte le unità di cibo è diverso nei due casi. La zona che presenta maggiori differenze è quella relativa ai risultati ottenuti simulando un mondo popolato da poche decine di agenti. In particolare con il nostro modello (Figura 2b) si ottengono risultati migliori soprattutto nella zona pochi agenti-poche unità di cibo.

La Figura 3 conferma questi stessi risultati anche nel caso di un approccio multirun. Come possiamo vedere il nostro modello anche in questo caso risulta essere più efficiente del flocking, dati pochi agenti (10) ed indipendentemente dalle unità di cibo presenti.

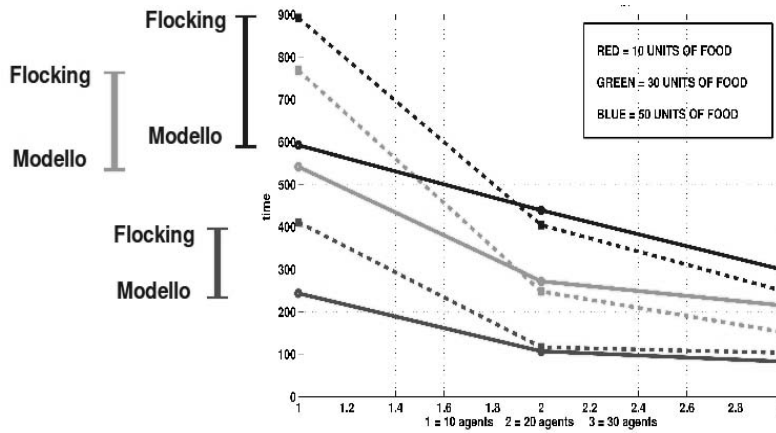



Figura 3. Le linee continue rappresentano i risultati ottenuti dal nostro modello. Le linee tratteggiate rappresentano i risultati ottenuti da un tradizionale algoritmo di flocking. Il nostro modello presrnta prestazioni migliore (minor tempo impiegato per trovare tutte le unità di cibo) quando il gruppo è composto da pochi agenti (10) per ognuna delle condizioni sperimentali relative alle unità di cibo (10, 30, 50); in questo caso abbiamo fatto variare in maniera più brusca il numero degli agenti (da 10 a 30 con step di 10) e quello di unità di cibo presenti nel mondo (da 10 a 50 con step di 20), e ripetuto le simulazioni con lo stesso set di parametri per diversi runs (20); quindi abbiamo preso i valori medi.

## *4. Conclusioni*

I risultati ottenuti con entrambi gli approcci sono particolarmente interessanti date certe condizioni.

Infatti, sopra una certa soglia relativa al numero di agenti (# agenti > 30), i due modelli (il nostro e quello di flocking) risultano sostanzialmente equivalenti.

I risultati interessanti si ottengono, quindi, simulando un mondo in cui pochi agenti (poche decine) devono trovare il cibo sparso a macchia di leopardo nell'ambiente.

Questo risultato, seppur preliminare, sembra fornire incoraggianti indicazioni rispetto a quello che avviene in natura per i grandi mammiferi carnivori (come lupi e leoni), che si organizzano rispettivamente in gruppi (pack per i lupi e pride per i leoni) di massimo 10 e 30 individui.

Pensiamo che le dinamiche di foraging nelle popolazioni di mammiferi carnivori possano essere studiate utilizzando il nostro modello in modo utile poichè esso considera apprendimento individuale, apprendimento collettivo e velocità e tutti questi elementi sono sicuramente importanti nei comportamenti di foraging in natura.

Al di là delle specifiche strategie di caccia adottate dagli animali, il modello sembra cogliere alcune caratteristiche (derivanti da pressioni evolutive) che hanno portato determinate specie ad organizzarsi socialmente in gruppi di piccole dimensioni.

Futuri studi orientati all'implementazione di strutture sociali (come gerarchie e presenza di un maschio alfa) ed abilità cognitive più complesse (come vere e proprie strategie predatorie) potranno fornire ulteriori elementi circa l'utilità del modello proposto.

**Bibliografia**

1. Caraco, T. and Wolf, L. L. (1975) Ecological Determinants of Group Sizes of Foraging Lions, The American Naturalist, Vol. 109, No. 967, pp. 343-352.
2. Nudds, T. D. (1978) Convergence of Group Size Strategies by Mammalian Social Carnivores, The American Naturalist, Vol. 112, No. 987, pp. 957-960.
3. Caraco, T. (1980) On Foraging Time Allocation in a Stochastic Environment, Ecology, Vol. 61, No. 1, pp. 119-128.
4. MacArthur, R. H. and Pianka, E. R. (1966). On the optimal use of a patchy environment. American Naturalist, 100, 603-609.
5. Emlen, J. M. (1966). The role of time and energy in food preference. American Naturalist, 100, 611-617.
6. Bäck, T. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University, New York, 1996.
7. Kennedy, J., Eberhart, R. and Shi, Y. Swarm Intelligence. Morgan Kaufman, 2001.
8. Rodman, P. S. (1981) Inclusive Fitness and Group Size with a Reconsideration of Group Sizes in Lions and Wolves, The American Naturalist, Vol. 118, No. 2, pp. 275-283.
9. Clark, C. W. and Mangel, M. (1984) Foraging and Flocking Strategies: Information in an Uncertain Environment, The American Naturalist, Vol. 123, No. 5, pp. 626-664.

10. Fonseca, C. and Fleming, P. An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computation, vol 3, no 1:1-16, 1996.
11. Nolfi, S. and Floreano, D. Evolutionary Robotics, Cambridge, MA: MIT Press. 2000.
12. Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, 1989.
13. Taha H. A. Operations research. New Dlhi, 2005.
14. Patel, M., Honavar, V. and Balakrishnan, K. (Ed) Advances in the Evolutionary Synthesis of Intelligent Agents. Cambridge, MA: MIT Press. 2001.
15. Hu, X., Eberhart, R. C. and Shi, Y. Particle swarm with extended memory for multi-objective optimization. In Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, 2003.
16. Kennedy, J. and Eberhart, R. Particle swarm optimization, in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.
17. Engelbrecht, A. P. Fundamentals of Computational Swarm Intelligence. Wiley, 2005.
18. PSO–wikipedia [http://en.wikipedia.org/wiki/Particle_swarm_optimization]
19. Parsopoulos, K. E. and Vrahatis, M. N. Particle swarm optimization method in multi-objective problems. In Proceedings of the 2002 ACM symposium on applied computing, 2002, 2002.
20. Ravindran, A., Phillips D.T., and Solberg, J. J. Operations Research - Principle and practice. John Wiley & Sons, New York, 2001.
21. http://www.stigmergicsystems.com/stig_v1/stigrefs/article1.html
22. http://ccl.northwestern.edu/netlogo/
23. Wolpert, D. H., Macready, W. G. (1997), No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation* 1, 67. [http://ic.arc.nasa.gov/people/dhw/papers/78.pdf]
24. http://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization

# *Routing su Mobile Ad Hoc Networks con Algoritmi Ispirati dal Comportamento Collettivo di Insetti*

**Filomena de Santis**
*Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"*,
*Università di Salerno*;
Via Ponte don Melillo
84084 Fisciano (SA) , Italy
Tel. +39 089969724, Fax +39 089969600
fds@unisa.it

## *1. Introduzione*

Il continuo miglioramento nel campo delle tecnologie di comunicazione ed il relativo aumento nella richiesta di connessioni ha comportato lo sviluppo massiccio di reti di calcolatori con fili e senza fili. Queste reti sono caratterizzate dal fatto di essere grandi o molto grandi, fortemente eterogenee dal punto di vista delle tecnologie di comunicazione, dei protocolli e dei servizi ed estremamente dinamiche a causa della variabilità della topologia, delle condizioni di traffico e del numero di utenti e servizi richiesti istante per istante. Tutto ciò ha reso necessario che gestione, controllo e fornitura di servizi diventassero di tipo 'intelligente' ed 'autonomino', tanto da implicare la definizione di nuovi protocolli ed architetture di rete.

In questo contesto particolare importanza ha assunto il routing che deve mantenere il passo con la complessità, l'eterogeneità e la dinamicità delle reti, scegliendo il miglior cammino tra due host qualsiasi in maniera autonomica, distribuita e dinamica sulla base della costante evoluzione dello stato della rete.

La letteratura nel campo del routing è molto ampia poiché gli algoritmi relativi hanno seguito una vita parallela allo sviluppo della tecnologia delle reti e all'evoluzione della domanda da parte degli utenti. Intorno agli anni 90 sono stati presentati i primi algoritmi ispirati da processi osservati nelle società di insetti. Queste ultime, infatti, sono caratterizzate dalla presenza di un insieme di unità distribuite, autonome e minimaliste che, attraverso interazioni locali si auto-organizzano, per

produrre un comportamento complessivo del sistema intelligente, adattabile ai cambiamenti dell'ambiente esterno, robusto rispetto a possibili danneggiamenti o perdite di unità, e fortemente scalabile a causa della modularità e dell'organizzazione distribuita. Tra le varie reti di calcolatori, che presentano caratteristiche equivalenti e, al contempo, beneficiano fortemente di algoritmi di routing basati su un numero molto alto di controllori autonomi e completamente distribuiti, saranno in questo lavoro considerate le MANET (Mobile Ad Hoc Networks).

## *2. MANET*

Le reti mobili senza fili appartengono a due classi distinte [1]: quella delle infrastrutturate, con gateways fissi e cablati, e quella delle non-infrastrutturate, le cosiddette reti ad hoc. Le prime funzionano con l'ausilio di una stazione di base: un'unità mobile si connette e comunica con la stazione di base più vicina all'interno del suo raggio di comunicazione cosicchè, quando, essa si sposta è necessario un handoff con un'altra stazione di base affinchè il processo di comunicazione possa continuare senza interruzioni. Le seconde non hanno routers fissi; tutti i nodi si possono muovere e si possono connettere arbitrariamente. Ogni nodo in una MANET non solo si comporta come host, ma anche come un router che trova e mantiene rotte verso tutti gli altri nodi della rete. Le MANET, a causa della mobilità dei nodi, si auto-organizzano e auto-configurano di volta in volta che la struttura della rete cambia; i nodi utilizzano lo stesso canale ad accesso casuale senza fili cooperando per realizzare un inoltro multihop. Data la mancanza di una infrastruttura di supporto e la possibilità di inoltro verso un host di destinazione, che è fuori del range del nodo sorgente, è sempre necessaria una procedura di routing per trovare un cammino attraverso cui inoltrare in maniera appropriata i pacchetti tra il nodo sorgente e quello destinazione. I problemi più consistenti che il routing nelle MANET presenta possono essere sintetizzati come segue.

'Collegamenti asimmetrici': la maggior parte delle reti cablate lavora con collegamenti simmetrici che sono sempre fissi. Ciò non vale nelle reti ad-hoc poiché i nodi sono mobili e cambiano continuamente la propria posizione nella rete causando situazioni problematiche quando, ad esempio, un nodo A invia un segnale ad un nodo B senza informazione alcuna sulla connessione nella direzione inversa.

'Sovraccarico nelle tavole di routing': dato l'alto tasso di mobilità dei nodi, le tavole di routing restano spesso sovraccaricate da rotte obsolete.

'Interferenza': a causa dei collegamenti altamente variabili in funzione delle carattteristiche della trasmissione, ogni trasmissione può interferire con un'altra e ogni nodo può ascoltare trasmissioni di altri nodi corrompendo l'intero processo di comunicazione.

'Dinamicità della topologia': poiché cambiano molto velocemente sia la posizione del nodo che le caratteristiche del mezzo di trasmissione, le tavole di routing devono tener conto di questi cambiamenti ed i relativi algoritmi di routing si devono riadattare. Valga ad esempio il fatto che nelle reti cablate l'aggiornamento delle tavole di routing avviene tipicamente ogni 30 secondi, laddove nelle MANET la frequenza di aggiornamento diventa notevolmente più alta.

## *3. Protocolli di Routing per MANET Ispirati da Colonie di Formiche*

I comportamenti di classi di intelligenze collettive, quali api e formiche, sono stati massicciamente utilizzati nella progettazione di algoritmi per MANET poiché essi garantiscono vantaggi come scalabilità, tolleranza ai guasti, adattività, velocità, modularità, autonomia e parallelismo. Più precisamente, sono state utilizzate la capacità delle colonie di formiche di scoprire i cammini più corti tra il proprio nido e le sorgenti di cibo e le strategie di comunicazione e gestione delle scorte adottate dalle api negli alveari.

Per ciò che concerne le colonie di formiche, è stato osservato che, mentre esse sono in movimento alla ricerca di cibo, lasciano sul proprio sentiero tracce di feromone e, ad ogni passo, decidono localmente di avanzare verso le zone adiacenti segnate da una maggiore intesità di feromone. I cammini più corti tra il nido e la sorgente di cibo possono essere completati più velocemente e più frequentemente da formiche che vanno avanti e indietro, aumentando in questo modo l'intensità di feromone. Questi cammini attraggono nel tempo una quantità di formiche sempre maggiore, che a loro volta, aumentano i livelli di feromone su quei cammini fino a che non si converge alla situazione in cui la maggioranza delle formiche si trova sul cammino più breve. L'intensità locale del feromone codifica una misura di bontà spazialmente distribuita che è associata localmente ad ogni decisione di movimento. Essa è il risultato del campionamento ripetuto e concorrente che le formiche effettuano del territorio; in effetti, essa è il risultato di un processo col-

lettivo di rinforzo dell'apprendimento che avviene a livello dell'intera colonia. Questa forma di apprendimento e controllo distribuito, basato sulla comunicazione indiretta tra gli agenti (le formiche), che localmente modificano l'ambiente e reagiscono a queste modificazioni portando ad una fase di coordinazione globale delle azioni degli agenti, è nota come 'stigmergia'. La coordinazione stigmergica è uno degli strumenti fondamentali per ottenere comportamenti auto-organizzanti non solo nelle colonie di formiche, ma anche in altri sistemi sociali. Quando è presente la stigmergia i protocolli del sistema giocano un ruolo predominante rispetto agli agenti stessi, che possono essere relativamente molto semplici. Un buon modello stigmergico fornisce robustezza, scalabilità ed evolvibilità.

La capacità delle colonie di formiche di risolvere problemi di cammino minimo in maniera distribuita, usando un numero di agenti minimalisti e una comunicazione stigmergica mediata dal feromone, ha prodotto la meta-euristica ACO (Ant Colony Optimization) utilizzata per la soluzione generalizzata e distribuita dei problemi di cammino minimo su grafi e, quindi, su reti. Le caratteristiche della meta-euristica ACO sono: costruzione di un cammino attraverso un sistema distribuito di agenti leggeri, uso di una strategia di decisione stocastica per costruire incrementalmente un cammino da un agente che si sposta passo dopo passo da un nodo del grafo ad un nodo adiacente, comunicazione stigmergica tra gli agenti attraverso variabili locali al nodo (variabili del feromone), apprendimento collettivo stigmergico delle variabili di feromone, che codificano la qualità attesa di ciascuna decisione sul prossimo nodo da includere nel cammino che si sta costruendo.

L'applicazione della meta-euristica ACO al routing sulle MANET deriva immediatamente dal fatto che la definizione di cammini di routing ottimali in un ambiente di rete può essere configurata come una particolare istanza del problema del cammino più corto dove i pesi degli archi sono valori dinamici che dipendono dalla larghezza di banda, dal ritardo di propagazione e dal traffico in input [2, 3].

## *4. Protocolli di Routing per MANET Ispirati da Colonie di Api*

Più recentemente, anche le colonie di api sono state oggetto di studio come potenziale sorgente di ispirazione per la progettazione di strategie di ottimizzazione per problemi dinamici e multi-obiettivo. Le colonie di api mostrano caratteristiche strutturali simili a quelle delle formiche, quali la presenza di una popolazione di individui sociali minima-

listi e devono affrontare problemi analoghi, quali il nutrimento distribuito, la costruzione del nido e il relativo mantenimento. Le api usano un sofisticato protocollo di comunicazione che consente loro di comunicare direttamente attraverso segnali del tipo ape-ape e, se necessario, stigmergicamente del tipo gruppo-ape o ape-gruppo. La stigmergia viene raggiunta attraverso una forma di comunicazione visuale che gioca il ruolo equivalente del feromone. I principali meccanismi di lavoro nelle colonie di api utilizzati per la progettazione di algoritmi di routing sono tre: la divisione adattiva ed età dipendente del lavoro, la comunicazione all'interno della colonia e il reclutamento dei lavoratori, la selezione stocastica dei siti di cibo. Il primo di questi meccanismi, che nella norma vede le api più giovani (una settimana) impegnate in apprendimenti di base, le api di mezza età (due o tre settimane) impegnate in compiti di mantenimento, quali secrezione della cera e lavorazione del nettare, e le api più anziane (più di tre settimane) impegnate nella ricerca del cibo e la difesa del nido, è altamente adattivo nel senso che le precedenti fasi possono essere velocemente modificate in caso di alterazione delle condizioni della colonia. Il secondo meccanismo risulta essere completamente distribuito ed effettuato in maniera competitiva: le api cercatrici di cibo annunciano una fonte di cibo attraverso la' waggle dance' la cui funzione è quella di codificare la direzione e la distanza della fonte. La 'waggle dance' è una forma diretta di comunicazione agente-agente. Il terzo meccanismo, infine, consente alle api non ancora impegnate nella ricerca di cibo di scegliere, nel caso più sciami indicano più fonti di cibo con più 'waggle dance', una indicazione tra quelle sulla base di una regola stocastica. In conseguenza di ciò, la colonia distribuisce la ricerca di cibo su diverse fonti cosicchè, nel momento in cui una fonte è quasi esaurita, c'è già una parte della colonia che sta esplorando un altro sito.

L'applicazione di meta-euristiche derivanti dal comportamento delle api nel routing su MANET ha consentito di sviluppare algoritmi in cui fosse presente un effettivo bilanciamento tra esplorazione e vantaggio[2, 3] .

## *5. Conclusioni*

Il problema del routing per reti della attuale e della prossima generazione è caratterizzato dal fatto di essere molto complesso, dinamico, eterogeneo e di grandi dimensioni. La 'swarm intelligence' può aiutare nelle sfide poste da queste reti in quanto presenta una serie di caratteristiche e proprietà altamente desiderabili nel contesto delle reti in gene-

rale e delle MANET in particolare. Un aspetto fondamentale della 'swarm intelligence' è l'approcccio alla progettazione di tipo 'bottom-up' che per il routing su reti comporta la definizione di protocolli che, tra le varie caratteristiche, presentano comportamenti localmente interattivi e auto-organizzanti, disponibilità di cammini di routing multipli e backup di fallimento, adattamento rapido e robusto rispetto a variazioni della topologia, del traffico e del deterioramento dei componenti, scalabilità e semplicità di progettazione e monitoraggio.

## *Bibliografia*

1. Abolhasan, M., Wysocki, T., Dutkiewicz, E.: A Review of Routing Protocols for Mobile Ad Hoc Networks. Ad Hoc Net.. 2, 1--22 (2004)
2. Ducatelle, F., Di Caro, G.A., Gambardella, L.M.,: An Evaluation of Two Swarm Intelligence MANET Routing Algorthms in an Urban Environment. In: 2008 IEEE Swarm Intelligence Symposium, St Louis MO USA (2008)
3. Farooq, M., Di Caro, G.A.,: Routing Protocols for Next Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview. In: Swarm Intelligence: Introduction and Applications, Springer, Natural Computing Series (2008)

# *Quando un insieme di reazioni è autocatalitico?*

**Alessandro Filisetti**
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
alessandro.filisetti@ecltech.org

**Roberto Serra**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative*,
*Università di Modena e Reggio Emilia;*
Via Allegri, 9 42100 Reggio Emilia;
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
roberto.serra@unimore.it

**Marco Villani**
*Dipartimento di Scienze Sociali, Cognitive e Quantitative*,
*Università di Modena e Reggio Emilia*;
Via Allegri, 9 42100 Reggio Emilia;
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
marco.villani@unimore.it

**Timoteo Carletti**
*Dipartimento di Matematica, Facoltà Universitaria Notre Dame de la Paix*;
rempart de la Vierge 8, B 5000 Namur, Belgium;
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
timoteo.carletti@fundp.ac.be

**Rudolf Marcel Füchslin**
*Laboratorio di Intelligenza Artificiale di Zurigo*;
Andreasstr. 15 CH-8050 Zürich, Switzerland; Tel. +41 (0)44 635 45 91
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
fuchslin@ifi.uzh.ch

**Irene Poli**
*Dipartimento di Statistica, Università Ca' Foscari*;
San Giobbe - Cannaregio 873, 30121 Venezia
*European Centre for Living Technology*;
S. Marco 2940, 30124 - Venezia
irenpoli@unive.it

## *1. Introduzione*

L'emergenza di uno o più cicli auto-catalitici all'interno di una rete di molecole interagenti è una proprietà fondamentale sia nello sviluppo di possibili scenari legati all'origine della vita, sia nell'indirizzare la ricerca di laboratorio verso lo sviluppo di nuove molecole capaci di evolversi interagendo con i propri bersagli.

Alcuni modelli teorici di reti catalitiche hanno dimostrato una certa predisposizione alla comparsa di cicli, fenomeno che al contrario difficilmente si riesce ad ottenere nei laboratori. Uno studio approfondito dei fenomeni in questione potrebbe aiutare a colmare il gap tra la teoria e i dati sperimentali.

La vita che conosciamo oggi è il risultato di miliardi di anni di evoluzione iniziati con le prime protocellule in grado semplicemente di duplicarsi mantenendo al loro interno il contenuto informativo necessario alla replicazione. Le prime forme di vita erano quindi molto più semplici di quelle attuali ma un minimo livello di complessità fu comunque necessario.

Già nei lavori di Eigen (1979) viene messa in evidenza l'importanza dei cicli autocatalitici nello sviluppo della vita poiché, in assenza di cicli, molecole troppo lunghe, a causa della mancanza di dispositivi di controllo, sarebbero andate incontro a troppi errori di replicazione, mentre molecole troppo corte non sarebbero state in grado di duplicarsi.

Ci troviamo in presenza di un ciclo autocatalitico quando ogni membro del ciclo è il prodotto di almeno una reazione catalizzata da almeno un altro membro del ciclo, Farmer et al. (1987).

Il nostro studio prende spunto dai lavori di Stuart Kauffman (1986) e Farmer et al. (1987) nei quali è stato sviluppato un modello contenente due tipi di reazioni (condensazione e cleavage) ed in cui tali reazioni vengono catalizzate dalle altre molecole presenti nel sistema. Il risultato fondamentale dei loro lavori è che aumentando la lunghezza massima delle molecole presenti il numero di reazioni possibili cresce più velocemente del numero di specie molecolari possibili, rendendo così inevitabile la comparsa di cicli.

L'obiettivo del nostro lavoro è di migliorare il modello originale introducendo una dinamica stocastica delle molecole, basata sul noto algoritmo di Gillespie, in modo da poter trattare adeguatamente i problemi connessi alla numerosità degli esemplari delle varie specie mole-

colari, che in alcuni casi può essere anche molto bassa, e alla cinetica delle reazioni.

Introducendo nel sistema una dinamica asincrona diventa però cruciale capire quale sia il grafo di reazioni su cui verificare l'eventuale presenza di cicli autocatalitici. Infatti, poiché le reazioni non sono simultanee, una particolare reazione teoricamente possibile potrebbe non avvenire per un tempo molto lungo, ad esempio per motivi cinetici. Nel seguito introdurremo due fra i possibili tipi di grafi di reazione, e mostreremo come la presenza o meno di cicli dipenda in maniera cruciale, e per certi aspetti non intuitiva, dalle scelte fatte. L'importanza di queste considerazioni appare evidente alla luce del fatto che la ricerca di condizioni per l'autocatalisi è uno degli obiettivi principali di questi modelli.

## *2. Il modello*

L'alto grado di complessità che caratterizza un sistema biologico composto da un grande numero di molecole interagenti rende necessaria una serie di semplificazioni.

Il modello utilizzato in questo lavoro è composto da catene unidimensionali orientate da sinistra verso destra, formate da caratteri appartenenti ad un alfabeto arbitrario di $n$ caratteri. Non formuleremo ipotesi sulla natura chimica delle molecole (p.es. Polipeptidi, acidi nucleici o lipidi) mantenendo la trattazione ad un elevato livello di astrazione.

Le specie sono caratterizzate da lunghezza, sequenza e dal numero di molecole appartenenti a esse.

Come nel modello originale proposto da Kauffman (1986), due sono le reazioni possibili: una reazione di condensazione nella quale due specie vengono unite, ed una reazione di cleavage nella quale, al contrario, due specie vengono formate dalla scissione di un'altra molecola.

Entrambe le reazioni per avvenire richiedono l'intervento di un catalizzatore.

Un'assunzione del modello è che la cinetica delle reazioni catalizzate sia molto più veloce di quella delle reazioni spontanee ed è questo il motivo per cui queste ultime non vengono prese in considerazione, inoltre assumeremo che le reazioni catalizzate siano irreversibili ovvero

che la velocità della reazione inversa sia talmente piccola da poter essere trascurata.

Il modello è logicamente diviso in due parti: l'inizializzazione e la dinamica. Durante l'inizializzazione viene creata la popolazione iniziale utilizzando come parametri la variabilità dell'alfabeto e la lunghezza massima iniziale delle molecole. Fatto questo viene calcolato il numero totale di reazioni possibili, di cleavage e condensazione, considerando le specie presenti, ed in base alla probabilità che avvenga una reazione ogni specie catalizzerà un certo numero di reazioni.

La simulazione viene eseguita utilizzando l'algoritmo stocastico di Gillespie (1977) attraverso il quale ad ogni step è possibile calcolare sia la reazione, sia l'intervallo temporale entro il quale questa avviene.

Si assume inoltre che la reazione avvenga in un reattore ben miscelato e che le concentrazioni di tutte le specie siano costanti in tutto lo spazio.

Si noti che, mentre una reazione di cleavage coinvolge due specie molecolari, il substrato ed il catalizzatore, una reazione di condensazione coinvolge tre specie molecolari, il catalizzatore e due substrati. Per tenere in considerazione questa caratteristica la reazione di condensazione avviene in due momenti, nel primo il catalizzatore lega il substrato formando un complesso molecolare, mentre nel secondo il complesso molecolare incontra il secondo substrato terminando la reazione e rilasciando il prodotto. Nel caso del cleavage invece la reazione avviene istantaneamente nel momento dell'incontro tra catalizzatore e substrato.

Al momento non vi è un legame funzionale tra la sequenza e la reazione catalizzata, perciò una qualsiasi molecola può catalizzare la formazione di diversi prodotti nello spazio delle sequenze ed è altresì vero che la formazione di una specie può essere catalizzata da più molecole nello spazio delle sequenze.

L'architettura del modello permette inoltre che le reazioni possano essere inibite poiché una specie, che per una determinata reazione funge da catalizzatore, potrebbe servire da substrato per un'altra e per questo potrebbe ad esempio essere scissa, scomparendo così dal sistema. È inoltre presente una competizione per le risorse che porta a diminuire la velocità di alcune reazioni.

### *3. Creazione del grafo dinamico delle reazioni*

È possibile definire diversi grafi di reazione per un sistema come quello sopra descritto, sia per quanto riguarda la natura dei nodi che l'insieme di reazioni. In relazione al primo aspetto, in questo lavoro ci limiteremo a considerare reti i cui nodi sono i catalizzatori, e in cui esiste un link orientato da A a B se il primo catalizza la formazione del secondo. B potrebbe semplicemente essere il prodotto della reazione ma, allo stesso tempo, potrebbe anche essere l'origine di un nuovo link se catalizzasse anch'esso la produzione di altre molecole.

Ricordiamo che intendiamo studiare l'evoluzione del sistema nel tempo, a partire da un insieme limitato di specie iniziali. Quindi un tipo di grafo molto importante è quello delle cosiddette reazioni possibili ad un dato istante (G1 nel seguito), composto da tutte le reazioni possibili fra le molecole presenti nel sistema all'istante *t*. In un certo senso, questo grafo descrive il futuro prossimo possibile del sistema (nearest adjacent possible, secondo la terminologia di Kauffman (2008)).

Si noti che in questo grafo una reazione scompare istantaneamente non appena scompare uno dei costituenti, che pure potrebbe essere ricostituito pochi istanti dopo. Questo effetto è problematico in un sistema asincrono, in cui il time step è molto breve. Un altro aspetto problematico è legato al fatto che vi possono essere reazioni assai improbabili, ad esempio perché le costanti cinetiche sono molto piccole, e che quindi non avvengono con frequenza apprezzabile. Queste reazioni compaiono comunque nel grafo G1.

Per questi motivi prenderemo in considerazione anche un secondo grafo, quello delle reazioni attuali (G2), che comprende solo le reazioni che sono avvenute almeno una volta dall'istante iniziale. Una reazione resta nel grafo finché i substrati e i catalizzatori permangono, mentre quando uno di essi sparisce la reazione resta per un periodo limitato. Se in questo intervallo essa non avviene nuovamente, oppure non ricompaiono i costituenti necessari, la reazione viene eliminata.

Per quanto riguarda l'individuazione dei cicli abbiamo considerato due metodi. Il primo è approssimativo, Farmer et al. (1986), e si basa sul calcolo del rapporto *r* tra archi e nodi. Il secondo metodo riguarda invece le proprietà della matrice di adiacenza dei catalizzatori, Jain (1998, 2001 e 2002). Consideriamo una matrice quadrata con dimensione pari al numero dei catalizzatori presenti nel grafo in questione. Se vi è un link diretto tra due catalizzatori il relativo elemento della matrice sarà 1, altrimenti sarà 0. Secondo il teorema di Perron-Frobenious, dato che la matrice è reale e positiva, l'autovalore con la parte reale maggio-

re *(λ1)* sarà reale e non negativo. Si è dimostrato che un ciclo autocatalitico è presente quando $\lambda 1 >= 1$.

La Figura 1 mostra due comportamenti opposti relativi allo stesso sistema ed in cui l'unica variante riguarda l'insieme di reazioni scelte per la formazione del grafo. Osservando la figura di sinistra si nota che l'autovalore $\lambda 1$ va molto velocemente a zero indicando la repentina scomparsa di cicli autocatalitici all'interno del sistema. Osservando però la figura di destra si può osservare che invece l'autovalore e indice di connettività sono al di sopra della soglia utile per poter constatare la presenza di un ciclo autocatalitico nel grafo G2.

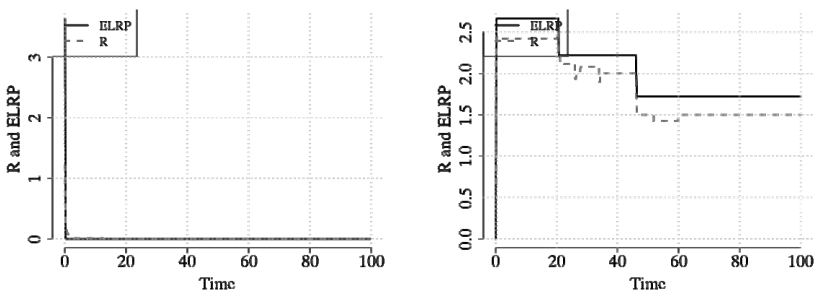



Figura 1: Il grafico di sinistra mostra il comportamento dell'autovalore (ELRP) e dell'indice di connettività (R) considerando il grafo G1. Il grafico sulla destra mostra invece il comportamento dell'autovalore e dell'indice di connettività nel grafo G2(tempo di vita in mancanza dei costituenti pari a 20 secondi).

Tale comportamento ci fa notare non solo l'importanza delle possibili reazioni che potrebbero avvenire in un dato momento, questo tipo di scelta da anche un'idea di cosa potrebbe avvenire in futuro, ma anche l'importanza della memoria che il sistema ha delle reazioni avvenute in precedenza.

## *4. Conclusioni*

In questo lavoro si è posta l'attenzione sulla creazione di un modello per lo studio dell'emergenza di cicli auto catalitici. Sono state presentate le caratteristiche principali del modello ed è stata posta l'attenzione

sull'importanza della scelta dell'insieme di reazioni per l'analisi del comportamento del sistema. Una prima indicazione riguardante il fatto che l'emergenza di tali cicli avviene solo a livello teorico e difficilmente viene riscontrata all'interno dei laboratori potrebbe essere racchiusa proprio nella scelta dell'insieme di reazioni volte a formare la rete catalitica. Fino ad oggi è stato preso in considerazione un grafo contenente tutte le reazioni possibili in un dato momento dando la stessa importanza a reazioni molto frequenti come a reazioni particolarmente rare o addirittura mai avvenute. Utilizzando al contrario l'insieme di reazioni che realmente accadono si è mostrata una tangibile differenza nel comportamento dei modelli.

Sebbene questo possa essere un primo passo nel colmare il gap tra teoria e risultati sperimentali rimangono diversi fattori da prendere in considerazione nel prossimo futuro tra i quali l'influenza delle specie presenti in piccola quantità, la possibile precipitazione di molecole lunghe e il ruolo giocato dall'energia.

**References**

1. D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry, 81(25):2340-2361, 1977.
2. M. Eigen and P. Shuster. The hypercycle: A principle of natural self-organization. Springer, Berlin, 1979.
3. S. A. Kauffman. Autocatalytic sets of proteins. J Theor Biol, 119(1):1-24, 1986.
4. J.D.Farmer, S. Kauffman, and N. H. Packard. Autocatalytic replication of polymers. Physica D, 22(2):50-67, 1986.
5. S. Jain and S. Krishna. Autocatalytic set and the growth of complexity in an evolutionary model. 1998.
6. S. Jain and S. Krishna. A model for the emergence of cooperation, interdependence, and structure in evolving networks. Proc Natl Acad Sci U S A, 98(2):543–547, 2001.
7. S. Jain and S. Krishna. Large extinctions in an evolutionary model: the role of innovation and keystone species. Proc Natl Acad Sci U S A, 99(4):2055-2060, 2002.
8. S. Kauffman: Reinventing the sacred, Basic Books, 2008.

# *Un modello basato sull'entropia per valutare algoritmi bio-ispirati*

**Gianluigi Folino**
*ICAR-CNR, Istituto per il Calcolo e le Reti ad Alte Prestazioni*;
Via P. Bucci, Cubo 41C, Rende (CS)
folino@icar.cnr.it

**Agostino Forestiero**
*ICAR-CNR, Istituto per il Calcolo e le Reti ad Alte Prestazioni*;
Via P. Bucci, Cubo 41C, Rende (CS)
forestiero@icar.cnr.it

## *1. Introduzione*

Colonie di formiche, sciami di api, termiti e stormi di uccelli possono essere modellati con sistemi ad agenti che esibiscono un comportamento collettivo intelligente (Swarm Intelligence) [1]. Tali algoritmi sono stati applicati con successo in tutta una serie di campi applicativi, dal problema generico della ricerca in uno spazio globale alla classificazione di documenti, al clustering, ecc..

Esistono poche metodologie per valutare la bontà di questi algoritmi nello svolgere un determinato compito. In questo lavoro è presentata e valutata una metodologia, introdotta per prima in [5], basata sul concetto di entropia, per valutare l'organizzazione presente negli algoritmi bio-ispirati e cercare di valutarne l'efficacia. Per illustrare meglio la metodologia, essa è stata studiata su un algoritmo di clustering bio-ispirato, basato su un flock di uccelli, presentato in [3]. Ad ogni modo tale metodo può essere generalizzato facilmente.

L'entropia può essere utilizzata per analizzare i comportamenti emergenti nei sistemi multi-agente. In tali sistemi, è possibile considerare un macro-livello in cui l'entropia diminuisce in contrapposizione a un microlivello in cui l'entropia aumenta. Il contributo fondamentale di questo lavoro è di mostrare che questo approccio basato sull' entropia può essere utilizzato per analizzare sperimentalmente la presenza di self-organization nel sistema stesso.

Il resto dell'articolo è organizzato come segue. La sezione 2 illustra brevemente l'algoritmo di flocking per il clustering approssimato che sarà analizzato. Nella sezione 3 è descritto il modello basato sull'entropia, che sarà poi valutato sperimentalmente nella sezione 4. La sezione 5 trae alcune conclusioni sul lavoro.

## *2. Un algoritmo di flock per il clustering approssimato*

L'algoritmo di flocking è stato proposto per la prima volta da Reynolds [6] e utilizzato come un metodo per la simulazione del flock di uccelli sia per animazioni che per lo studio dei comportamenti emergenti. Infatti, il flock è un esempio di comportamento collettivo emergente in cui non c'è un leader e non esiste un controllo globale, ma il comportamento emerge dall'interazione locale dei componenti (agenti). Il modello del flocking si basa su tre regole fondamentali:

(i) separazione, ogni agente deve mantenersi ad una data distanza dai propri vicini;

(ii) coesione, ogni agente non deve allontanarsi più di una certa distanza dai propri vicini;

(iii) allineamento, ogni agente deve mantenere la stessa direzione di spostamento del flock.

In [3] tale algoritmo è stato modificato, ispirandosi al lavoro presentato da Macgill [4], aggiungendo una strategia di *foraging* in modo da effettuare la ricerca, in modo vantaggioso, di elementi interessanti in dati spaziali e il clustering approssimato.

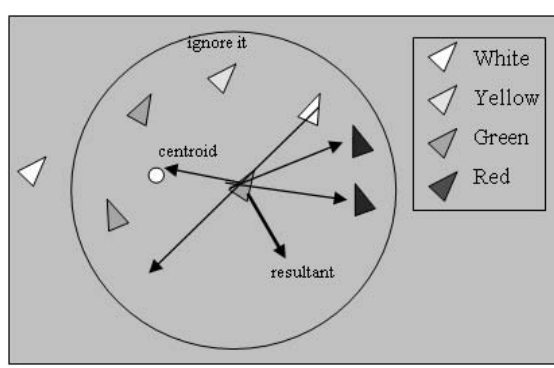



Figura 1. Calcolo della direzione di un agente di colore verde.

Più in dettaglio, ogni agente è colorato in base a caratteristiche locali, per esempio la densità dei dati attualmente presenti nel proprio raggio di visibilità. Agenti che stanno esplorando zone con un grado d'interesse differente avranno colori diversi: agenti rossi, indicheranno zone caratterizzate da dati interessanti, agenti verdi, zone con dati di medio interesse, agenti gialli, zone i cui dati hanno un basso interesse e agenti bianchi, zone da evitare. L'algoritmo di Reynolds è stato esteso in modo da inglobare questo comportamento (fig. 1). Nel dettaglio: per l'allineamento e la coesione, gli agenti di colore giallo sono ignorati, dal momento che le zone che stanno visitando non sono particolarmente interessanti; gli agenti bianchi contribuiranno con una forza repulsiva, per indicare zone di scarso interesse.

```
for i=1 : : : MaxIterations
    foreach agent (yellow, green)
        age=age+1;
        if (age > Max Life)
            generate_new_agent();die();
        endif
        if (not visited (current_point))
            property = compute_local_property(current_point);
            mycolor= color_agent(property);
        endif
    end foreach
    foreach agent (yellow, green)
        dir= compute_dir();
    end foreach
    foreach agent (all)
        switch (mycolor){
            case yellow, green: move(dir, speed(mycolor)); break;
            case white: stop(); generate_new_agent(); break;
            case red: stop(); generate_new_close_agent();
```

Figura 2. Pseudo-codice dell'algoritmo di flocking modificato.

In Fig. 2 è riportato uno pseudo-codice dell'algoritmo con le principali funzioni svolte. Nell'algoritmo, denominato SPARROW, è stato utilizzato un approccio basato su DBSCAN [2] e quindi sulla densità, in

cui il colore dell'agente è deciso in base al numero di oggetti presenti nel proprio raggio di visibilità. Si rimanda a [3] per maggiori dettagli sull'algoritmo. Ogni agente, rispettando le regole dell'algoritmo di flocking modificato, esplora lo spazio alla ricerca di zone con un'elevata concentrazione di dati interessanti, cambiando colore in funzione del numero di oggetti osservati (densità).

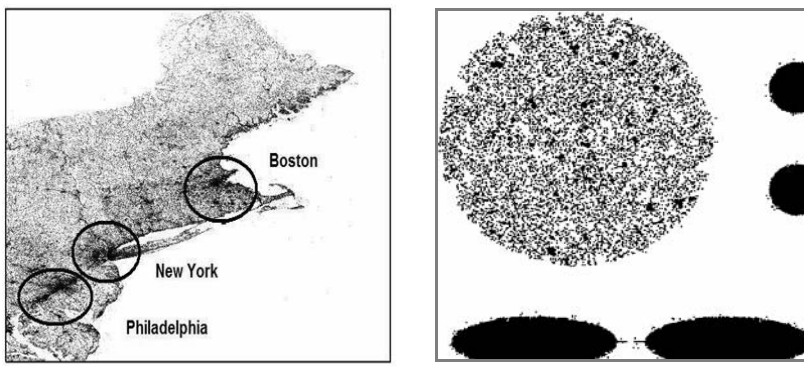



Figura 3. (a) North-East dataset e (b) CURE dataset

Nel momento in cui il proprio colore diventa rosso, cioè si trova in una zona interessante, si ferma e genera un nuovo agente in prossimità della zona osservata. Mentre, se il numero di oggetti osservati è pari a zero, il proprio colore diventa bianco, quindi si fermerà e genererà un nuovo agente in una posizione casuale dello spazio di ricerca. Gli agenti il cui colore è diventato rosso o bianco, si comporteranno rispettivamente da attrattore e da repulsore per gli altri agenti. In questo modo gli agenti saranno guidati verso le zone più interessanti ed eviteranno quelle povere di dati significativi.

## *3. Un modello basato sull'entropia per la valutazione di sistemi bio-ispirati*

Parunak e Brueckner [5] introducono un modello basato sull'entropia per analizzare i comportamenti emergenti nei sistemi multi-agente e sostengono che la relazione fra la self-organization e l'entropia può fornire una linea quantitativa e analitica per progettare e analizzare

sistemi multi-agente. Il risultato principale che si deduce riguarda il principio che per ridurre il disordine in un sistema multi-agente e raggiungere un comportamento globale coerente, bisogna accoppiarlo a un altro in cui il disordine incrementa. Questo corrisponde a un macro-livello in cui l'ordine cresce, mentre ci deve essere un micro livello in cui l'entropia aumenta.

Un sistema multi-agente segue la II legge della termodinamica, se gli agenti si muovono senza alcuna costrizione. Tuttavia, se è aggiunta informazione al sistema in modo intelligente, la tendenza naturale alla massima entropia del sistema sarà contrastata e il sistema si muoverò verso una situazione di *self-organization*.

Di seguito diamo una spiegazione più formale delle nozioni di entropia. Queste nozioni saranno poi applicate all'algoritmo SPARROW presentato nella sezione precedente, ma tutte le considerazioni possono essere facilmente adattate a tutta una serie di algoritmi di ricerca basati sulla swarm intelligence presenti in letteratura (formiche, sciami di api, ecc.).

Come enunciato in [5], possono essere osservati due livelli di entropia: uno macro in cui si manifesta organizzazione e uno micro in cui si ha un aumento di entropia. Nel nostro caso il micro è rappresentato dagli agenti bianchi e rossi che segnalano rispettivamente zone deserte e interessanti e il livello macro dal comportamento di tutti gli agenti. In effetti, noi ci aspettiamo un aumento dell'entropia dovuto alla nascita di nuovi agenti (bianchi e rossi) e una riduzione della macro-entropia dovuto al modello coordinato degli agenti. *Gli agenti, dopo un primo periodo in cui si muovono casualmente, si concentrano sulle zone interessanti per effetto dell'attrazione degli agenti rossi nel nostro caso (e per esempio per l'effetto del feromone nel caso delle formiche).*

Di seguito diamo delle definizioni formali. Nella teoria dell'informazione, l'entropia è definita come:

$$S = -\sum_i p_i \log p_i \tag{1}$$

dove $p_i$ rappresenta la probabilità che si verifichi un dato evento. Ora per adattare la (1) ai nostri scopi, si introduce un'entropia cosiddetta *locational*, perché si basa sulla posizione dell'agente, cioè sulla probabilità che ha l'agente di visitare una determinata cella, e non sul movimento. Consideriamo un agente che si muove in una griglia NxM=K, in cui tutte le celle hanno la stessa dimensione. Ogni agente avrà la stessa probabilità di cadere in una delle K celle della griglia. L'entropia può essere misurata sperimentalmente facendo girare l'algoritmo di flocking

per T volte e contando quante volte un agente cade nella stessa cella i per ogni iterazione. Dividendo questo numero per T, si otterrà la probabilità $p_i$ che l'agente si trovi in quella cella.
Quindi, l'entropia locational sarà:

$$S = -\sum_{i=1}^{K} \frac{p_i \log p_i}{\log K} \quad (2)$$

Il fattore di normalizzazione log k è dovuto al fatto che in caso di distribuzione casuale degli agenti ogni stato avrà probabilità 1/k e l'entropia totale sarà logk/logk=1.

L'equazione 2 può essere generalizzata per P agenti facendo la somma totale e dividendo per P. In pratica, essa rappresenta la macroentropia se noi consideriamo solo gli agenti bianchi e rossi, la microentropia se invece consideriamo tutti gli agenti.

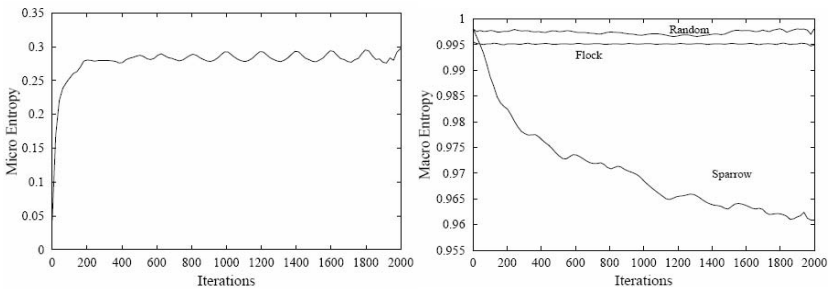
## *4. Risultati Sperimentali*

L'algoritmo di flocking è stato fatto girare per 100 volte per 2000 iterazioni usando 50 agenti e sono state calcolate la micro e macro entropia, utilizzando il dataset North-East (fig. 3a). Le due entropie sono mostrate nelle figure 4 a e b, con un confronto con l'algoritmo di flocking classico di Reynolds e con una disposizione casuale (random) degli agenti. Dalle figure si può osservare un aumento della micro entropia e una diminuzione della macro entropia dovuto al comportamento attrattivo degli agenti rossi e a quello repulsivo dei bianchi e quindi alla nascita di un'organizzazione. Al contrario, nel modello random e nel flock standard la curva della macro entropia è quasi costante, confermando l'assenza di organizzazione. La presenza di un'organizzazione da solo non basta a garantire che l'algoritmo esplori le regioni più interessanti dello spazio di ricerca. Per questo motivo una seconda serie di esperimenti considera il valore dell'entropia nelle zone in cui sono presenti cluster (fig. 5a) e nelle altre zone (figure 5b) per il dataset CURE mostrato in figura 3b.

CURE è più facile da analizzare per la separazione abbastanza netta fra i cluster rispetto a North-East.

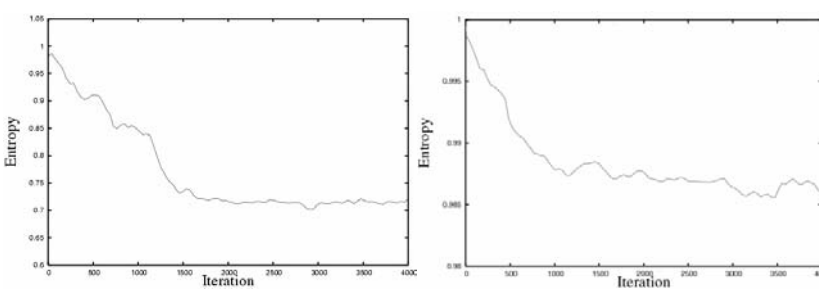
L'entropia, come si evince dalle figure, diminuisce sia nelle zone dense di cluster che in quelle deserte (perché gli agenti si tengono lontane da queste zone). Nelle zone di cluster si trovano circa l'80% dell'intero flock (mentre queste zone rappresentano solo il 65% dell'intero

spazio di ricerca). Questo conferma che non solo si rileva una presenza di organizzazione nell'algoritmo, ma questa organizzazione porta un maggior numero di agenti a presidiare le zone interessanti.




(a) (b)

Figura 4. (a) Macro entropia e (b) micro entropia per il dataset North-East.




(a) (b)

Figura 5. Macro entropia in zone (a) interne al cluster, (b) esterne al cluster per il dataset Cure.

## *5. Conclusioni*

In questo lavoro, è mostrato come una metodologia basata sul concetto di entropia può essere utilizzata per valutare algoritmi di ricerca bio-ispirati. In particolare questa metodologia è stata valutata su un algoritmo di clustering basato su un flock. Dagli esperimenti condotti su due datasets largamente usati, si rileva la presenza di self-organization,

attraverso il comportamento della macro entropia che decresce. Questo deve essere bilanciato dalla micro-entropia che ovviamente aumenta. Tuttavia, la presenza di organizzazione non assicura che la ricerca sia concentrata nelle zone realmente interessanti e quindi, un'ulteriore analisi deve essere condotta per verificare la bontà dell'algoritmo. Nel nostro caso questo è confermato da una maggiore presenza degli agenti nelle zone dei cluster.

## *Bibliografia*

1. Bonabeau, E., Dorigo, M., and Theraulaz, G. 1999. Swarm intelligence: from natural to artificial systems. Oxford University Press, NewYork, NY, USA.
2. Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
3. Folino G., Forestiero A., Spezzano G., An adaptive flocking algorithm for performing approximate clustering, Information Sciences, Elsevier, 2009, Vol. 179, n. 18, pp. 3059–3078.
4. Macgill J, Using flocks to drive a geographical analysis engine, in: M.A. Bedau, J.S. McCaskill, N.H. Packard, S. Rasmussen (Eds.), Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life, The MIT Press, Cambridge, Massachusetts, 2000, pp. 446–453.
5. Parunak, H. V. D. and Brueckner, S. 2001. Entropy and self-organization in multi-agent systems. In AGENTS '01: Proceedings of the fifth international conference on Autonomous Agents. ACM Press, New York, NY, USA, pp.124-130.
6. Reynolds, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In SIG-GRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques. ACM Press, New York, NY, USA.

# *Apprendimento supervisionato ad alte prestazioni su processore grafico con CUDA*

**Bernardino Frola**
*Dipartimento di Informatica "R.M. Capocelli", Università di Salerno*;
Via Ponte don Melillo - 84084 - Fisciano (SA);
Tel. +39 089969509, Fax +39 089969600
ber.frola@gmail.com

***Abstract***

Sfruttando la grande potenza di calcolo a disposizione dalle recenti GPU (Graphics Processing Unit), il nostro sistema si propone ridurre il tempo necessario per la valutazione dell'errore nell'apprendimento supervisionato delle reti neurali multilivello (MLP). I risultati ottenuti mostrano prestazioni fino a 30 volte maggiori rispetto alle comuni CPU.

## *1. Introduzione*

Negli ultimi anni, il rapido incremento in termini di performance delle GPU (Graphics Processing Unit) ha reso questo tipo di hardware un ottimo candidato all'utilizzo intensivo nel calcolo scientifico. Nel 2007 la NVIDIA ha introdotto CUDA (Compute Unified Device Architecture) [10]: un compilatore ed un insieme di strumenti che ha semplificato notevolmente la programmabilità di questi processori, introducendo una interfaccia C-like.

Presentiamo in questo documento una tecnica per sfruttare la potenza di elaborazione dei processori grafici nell'ambito dell'apprendimento supervisionato delle reti neurali multilivello (MLP) feed-forward, con la conseguente possibilità di ridurre, ma non eliminare, il problema dei minimi locali che affligge la ricerca dei parametri ottimali della fase di learning.

## *2. Lavori correlati*

Una delle prime esperienze di utilizzo della GPU nel calcolo scientifico è stato effettuato da Larsen and McAllister [7] (2001) per la velocizzazione della moltiplicazione di matrici. Moreland and Angel [9] (2003) hanno realizzato una delle prime implementazioni della trasformata di Fourier (FFT) su processore grafico. Nel 2006 Govindaraju et al. [4] hanno presentato un algoritmo di ordinamento efficiente su GPU, con ulteriori sviluppi nel data mining. Cao et al. [3] (2006) hanno discusso, nel loro lavoro, una tecnica per il clustering scalabile su processore grafico basato su k-means. Per quanto riguarda le reti neurali, nel 2005, Bernhard and Keriven [1] hanno portato le spiking neural network(SNN) su GPU e, nel 2006, Luo et al. hanno realizzato su questo tipo di hardware una implementazione di reti SOM e MLP.

Jang et al. [5] (2007), hanno utilizzato CUBLAS [11], una implementazione della libreria BLAS (Basic Linear Algbra Subprograms), distribuita con l'SDK di CUDA, per creare una implementazione mista GPU-CPU di reti neurali MLP, impiegandole nel riconoscimento di testi. Nel 2008, Lahabar et al. [6] hanno presentato una tecnica per il pattern recognition su GPU.

## *3. Informazioni preliminari*

### *3.1 Architettura GPU*

Le GPU sono processori caratterizzati da una architettura parallela SIMD (Single Instruction Multiple Data), sempre affianate da una memoria RAM dedicata, chiamamata memoria video, separata da quella utilizzata dalla CPU.

Grazie alla dotazione di centinaia di processori, in grado di eseguire istruzioni parallelamente, e la spropositata banda passante offerta dalla memoria video, le GPU di ultima generazione sono in grado di generare un throughput teorico di 1 TFLOPS in singola precisione e 60 GFLOPS in doppia precisione.

### *3.2 Modello di programmazione CUDA*

Il modello di programmazione proposto da CUDA è basato sul concetto di thread. Tutti i thread lanciati in esecuzione eseguono la stessa

funzione, chiamata kernel. L'unica variante che contraddistingue l'esecuzione di questa funzione da parte dei diversi thread è costituita dai dati elaborati. In Figura 1 è riportato lo schema che rappresenta il modello di programmazione proposto da CUDA. I thread lanciati in esecuzione sono divisi in blocchi logici e, quelli appartenenti allo stesso blocco, possono comunicare tra loro utilizzando la shared memory, una regione di memoria estremamente veloce.

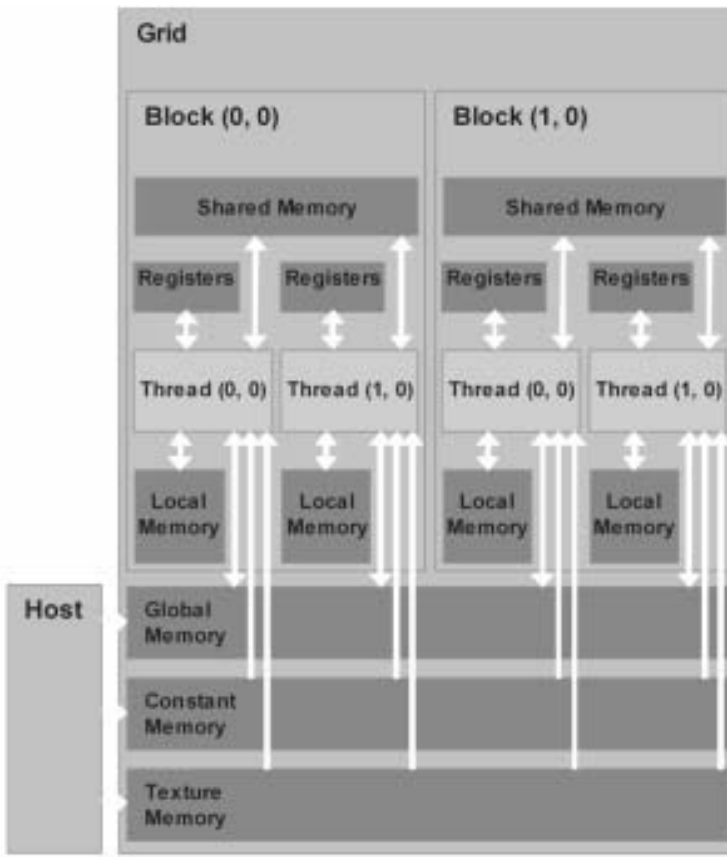



Figura 1: modello di programmazione CUDA [10]. Il componente Host indica la il processore e memoria RAM centrale.

Constant e texture memory sono due tipologie di memoria dotate di cache. Sono in grado di fare la differenza in termini di prestazioni, qualora ci sia la necessità di leggere ripetutamente gli stessi dati.
La CPU e la memoria centrale hanno la possibilità di scambiare dati con global, texture e constant memory solo prima o dopo il lancio del kernel. L'utilizzo della GPU è schematizzato di seguito:

1. Trasferimento dati di input dalla memoria CPU a quella GPU (global/texture/constant memory).
2. Lancio del kernel, specificando il numero di thread da eseguire.
3. Trasferimento dati di output dalla memoria GPU (global memory) verso la CPU

## *4. Modello proposto*

### *4.1 Definizione del problema*

Indichiamo con $x$ l'input costituito da $d$ dimensioni di una rete MLP a $L$ livelli. L'output della rete è indicato con $y$ ed è funzione dei pesi $w_1 \ldots w_L$ associati alle interconnessioni tra i neuroni, i valori di bias $b_1 \ldots b_L$, e l'input $x$. L'errore $E$ è definito come lo scarto quadratico medio tra l'output $y$ ed il target, ovvero, l'output previsto $t$ per lo specifico input $x$.

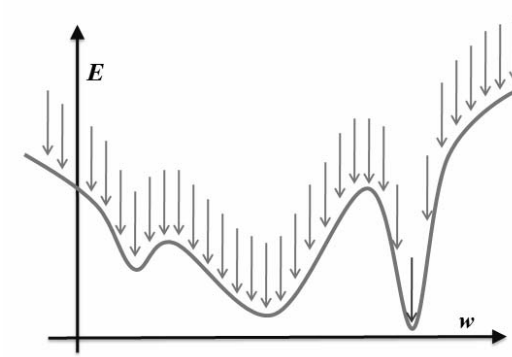



Figura 2: Funzione di errore $E$. Si tratta di una immagine semplificata, in quanto è riportata una singola dimensione dei pesi. L'utilizzo della tecnica del gradiente in casi del genere può portare a trovare minimi locali. Un campionamento fitto può aumentare le possibilità di trovare il minimo globale (in rosso).

L'apprendimento consiste nel fornire alla rete il training-set, costituito da un insieme di coppie $<x, t>$. Per ogni coppia vanno individuati i pesi e bias che minimizzano la funzione di errore $E$, poi utilizzati per l'aggiornamento di pesi e bias della rete che sta eseguendo l'apprendimento. L'algoritmo error back-propagation [2] è uno dei più comunemente utilizzati per la della ricerca di questi parametri. Il processo iterativo di questo algoritmo si basa sul concetto di grandiente della funzione di errore ed aggiorna i parametri della rete, procedento verso il valore minimo ritenuto ottimale. Questo metodo è efficiente ma in casi particolari, come quello illustrato in figura 2, è possibile incappare in minimi locali della funzione di errore.

### *4.2 Architettura proposta*

La tecnica proposta utilizza la potenza di calcolo della GPU per eseguire un elevatissimo numero di valutazioni di errore su diverse combinazioni casuali di pesi e bias.
Indichiamo con $N$, il numero dei campionamenti effettuati e con $T_1 \ldots T_N$ gli insiemi contenenti, ognuno dei quali, il set di pesi $w_1 \ldots w_L$ e bias $b_1 \ldots b_L$ da campionare, generati in modo casuale nella fase di Preparazione dati (figura 3).
Il calcolo dell'errore $e_1 \ldots e_N$ è effettuato in parallelo sulla GPU in due passi separati (figura 3), ognuno dei quali eseguito da un kernel CUDA:

- Passo I: valutazione dell'errore: viene generata un matrice $N * d$ composta da valutazioni di errore sulle singole dimensioni: $e_1^1 \ldots e_N^1$, $e_1^2 \ldots e_N^2$, $e_1^d \ldots e_N^d$. Sono lanciati in esecuzione esattamente $N * d$ thread CUDA, ognuno dei quali si occupa di una singola dimensione ed una singolo campionamento in $T_1 \ldots T_N$. La lettura dei dati di input $x$ avviene mediante l'utilizzo della texture memory. Utilizzando in modo opportuno la shared memory, pesi e bias sono letti da memoria in modo distribuito tra tutti i thread cui è stata assegnata la stessa dimensione.
- Passo II: somma su tutte dimensioni dei valori contenuti nella matrice generata al passo I, in modo da ricostruire le valutazioni di errore $e_1 \ldots e_N$: $e_1^1 + e_1^2 + \ldots + e_1^d$ , $e_2^1 + e_2^2 + \ldots + e_2^d$, … , $e_N^1 + e_N^2 + \ldots + e_N^d$. Sono lanciati $N$ thread CUDA, uno per ogni riga della matrice del passo I.

La fase di Elaborazione risultati (figura 3) si occupa di individuare la combinazione di parametri che garantisce l'errore minore, effettuando una semplice ricerca sull'output $e_1 \ldots e_N$. A questo punto è possibile utilizzare direttamente la configurazione associata all'errore minimo per l'aggiornamento del passo di apprendimento, oppure eseguire un ulteriore miglioramento, seguendo il gradiente della funzione di errore $E$.

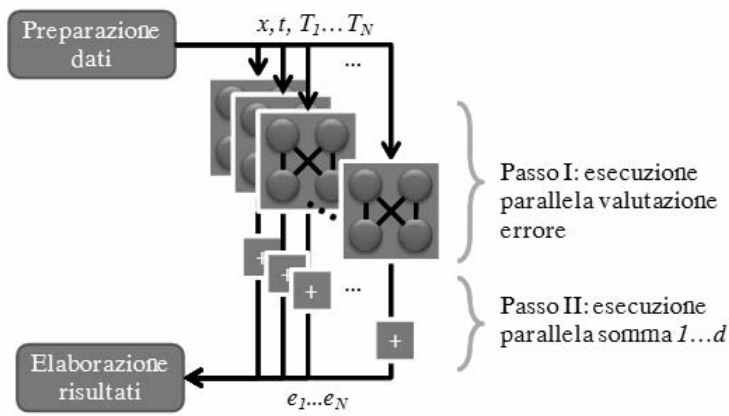



Figura 3: Schema di esecuzione parallela adottato. L'errore è valutato in modo parallelo su tutti gli elementi $T_1 \ldots T_N$.

## *5. Risultati ottenuti*

### *5.1 Prestazioni*

I risultati si riferiscono a reti neurali feed-forward a 2 e 3 livelli. Entrambe sono composte da 38 neuroni di input ed uno di output. La rete a 2 livelli è caratterizzata da 20 neuroni hidden. La rete a 3 livelli ha 15 neuroni hidden sul primo livello e 5 sul secondo. Il numero di dimensioni $d$ dei dati è pari a 1298.

Harware di riferimento: CPU Intel Core 2 2.0Ghz, 2Gb RAM, NVIDIA GTX285 1Gb RAM.

È stato utilizzato Matlab R2008, Windows XP SP3, CUDA v2.3. Tutti i test sono eseguiti su dati in virgola mobile a doppia precisione.

I valori ottenuti con la GPU sono stati contrapposti a quelli ottenuti con: CPU dual-core (hardware di riferimento); CPU quad-core

2xOpteron 240, 2Gb RAM. I risultati mostrati nel grafico in figura 4 mettono a confronto le due seguenti implementazioni:

- Matlab parfor: implementazione ad-hoc in Matlab, effettuando N valutazioni sequenziali, sfruttando il parallelismo offerto dalla istruzione parfor di Matlab (codice interpretato).
- Matlab GPU: implementazione ad-hoc Matlab-CUDA, utilizzando la tecnica di interfacciamento basata su MEX-FILE.

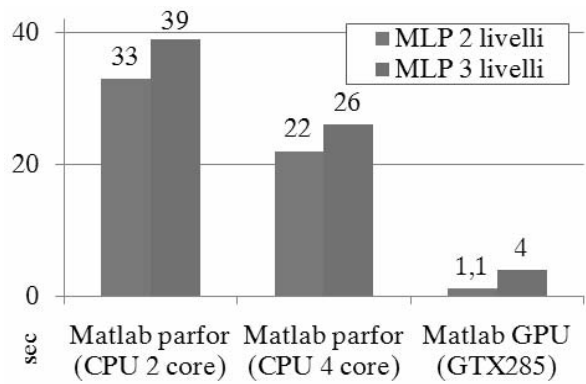



Figura 4: confronto tempi per la l'esecuzione di N = 10000 valutazioni di errore. Tutti i valori sono espressi in secondi.

### *5.2 Precisione di calcolo*

È bene sottolineare che l'utilizzo delle GPU in ambito del calcolo scientifico è afflitto da un problema parzialmente ancora attivo: la precisione di calcolo. I risultati proposti in questo documento riportano un errore relativo uguale a $10^{-16}$, causato dalla funzione di attivazione utilizzata (tanh), la cui implementazione su GPU non è ancora conforme allo standard IEEE754. Si tratta comunque di un notevole miglioramento rispetto a soluzioni basate sulle precedenti generazioni di processori grafici, in grado di garantire solamente la singola precisione, che implicherebbe un errore relativo pari a circa $10^{-6}$.

## *6. Conclusioni e lavori futuri*

Sfruttando la potenza di calcolo della GPU, il nostro sistema è in grado di ridurre i tempi di ricerca di pesi e bias nell'apprendimento supervisionato. Grazie al modello proposto, con una rete a 2 livelli, è possibile effettuare lo stesso numero di valutazioni di errore che effettua una CPU dual-core, in un tempo oltre 30 volte inferiore, e 20 volte inferiore rispetto ad una CPU quad-core. Utilizzando la GPU è possibile eseguire 100 milioni di valutazioni in circa 3 ore, contro le oltre 60 ore che sarebbero richieste utilizzando la CPU. Potendo eseguire più valutazioni nello stesso arco di tempo, risulta meno probabile, ma non impossibile, bloccarsi nei minimi locali della funzione di errore.
Per quanto riguarda i lavori futuri, il sistema proposto ha mostrato una scarsa scalabilità all'aumentare del numero di livelli della rete MLP a tre. Lo scarto con la CPU quad-core è sceso drasticamente, ottenendo prestazioni solamente 6 volte più veloci di quest'ultima. Questo risultato sottolinea la necessità di un lavoro di generalizzazione del sistema proposto, in modo da renderlo compatibile e totalmente performante con qualunque tipo di rete, qualunque sia il numero di livelli utilizzato.

## *Bibliografia*

1. Bernhard F., and Keriven R. Spiking Neurons on GPUs. ICCV06. 2006.
2. Bishop C. M. Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK. 1995.
3. Cao F., Tung A. K. H. and Zhou A. Scalable Clustering Using Graphics Processors. Lecture Notes in Computer Science, vol. 4016/2006, pp. 372-384, 2006.
4. Govindaraju N. K., Manocha D., Raghuvanshi N., Tuft D. Gpusort: High performance sorting using graphics processors. http://gamma.cs. unc.edu/ GPUSORT. 2006.
5. Jang H., Park A., Jung K. Neural Network Implementation using CUDA and OpenMP. DICTA '08.Digital Image, pp. 155-161. 2008.
6. Lahabar S., Agrawal P. , and Narayanan P. J., High Performance Pattern Recognition on GPU. *NCVPRIPG*. 2008.
7. Larsen E. S., and McAllister D. Fast matrix multiplies using graphics hardware. Proc. of ACM-IEEE conference on Supercomputing, 55-55. 2001.
8. Luo Z., Liu H. and Wu X. Artificial Neural Network Computation on Graphic Process Unit. Neural Networks, 2005. IJCNN '05. Proceedings. IEEE International Joint Conference on. 2005.

9. Moreland K. and Angel. E. The FFT on a GPU, Proceedings of SIGGRAPH Conference on Graphics Hardware, pp. 112-119, 2003.
10. NVIDIA. CUDA Programming Guide v2.1. 2008. http://developer. download.nvidia.com/compute/cuda/2_1/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.1.pdf
11. NVIDIA: NVIDIA CUBLAS Library, 2007. http://developer.download. nvidia. com/compute/cuda/1 0/CUBLAS Library 1.0.pdf.

# *Robotics Attack!*

**Onofrio Gigliotta**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia, 44 00185 Roma,
Tel. +39 0644595277, Fax +39 0644595243
onofrio.gigliotta@istc.cnr.it

**Valerio Sperati**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia, 44 00185 Roma,
Tel. +39 0644595277, Fax +39 0644595243
valerio.sperati@istc.cnr.it

**Stefano Nolfi**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia, 44 00185 Roma,
Tel. +39 0644595277, Fax +39 0644595243
stefano.nolfi@istc.cnr.it

## *1. Arrivano i robot, tanti robot*

Il dipartimento di statistica della federazione internazionale di robotica (IFR Statistical Department[1]) ha previsto che nel 2011 più di 18 milioni di robot popoleranno il nostro pianeta: una vera invasione ripresa dai media con preoccupazione. Il dato interessante, tuttavia, riguarda la proporzione di robot industriali e di robot di servizio. Questi ultimi, ovvero i robot progettati per migliorare la qualità della vita degli utenti, saranno nelle previsioni più di 17 milioni. Al contrario dei robot industriali fissati inesorabilmente e saldamente all'interno di fabbriche, i robot di servizio opereranno per le città e per le nostre case con svariati ruoli e differenti forme. In realtà molti di loro sono già presenti in ambiente domestico, il più famoso è l'aspirapolvere Roomba dell'azienda americana IRobot, ma non mancano tagliaerba, robot che puliscono le piscine e robot di intrattenimento come il fortunato Robosapien o il

[1] http://ifr.worldrobotics.org.

simpatico dinosauro Pleo (Fig. 1). La paventata invasione dei robot sarà quindi una marcia pacifica e allegra di robot che giocheranno un ruolo fondamentale nei servizi ed in particolare nel settore educativo. In questo ultimo campo, in particolare, ci aspettiamo l'avvento una robotica sottoforma di "materie prime" (sensori, servomeccanismi, microcontrollori etc.) assemblabili facilmente, ovvero senza alcuna nozione specialistica. Proprio come in un attacco d'arte, un attacco di robotica porterà gli utenti a creare in pochi semplici passi un robot e in altrettanti semplici passi sarà possibile dar loro un *cervello* o in altri termini un ricco bagaglio comportamentale in risposta a stimolazioni ambientali.

Figura 1. Da sinistra verso destra sono mostrati il robot tagliaerba, Romba l'aspirapolvere, il robot pulisci piscine, Robosapiens e Pleo il dinosauro.

### *2. Robotica di intrattenimento ed educativa*

Molti robot giocattolo presenti sul mercato, è il caso dei su citati Robosapien e Pleo ricadono sotto la categoria omnicomprensiva di robotica di intrattenimento, che racchiude tutti quei robot che hanno come principale target i bambini e che si presentano come naturale evoluzione della vecchia funzione del giocattolo e dell'animale domestico. Dove prima era la fantasia a far muovere il giocattolo o a immaginare relazioni affettive (per es. il peluche come classico sostituto di un animale domestico), adesso ci pensano servomeccanismi e sensori a regalare ai bambini esperienze di gioco più immersive. Benché l'industria dell'intrattenimento, nelle sue molte incarnazioni, rappresenti senza dubbio una grande opportunità di *business,* recludere la robotica per bambini al solo intrattenimento, tuttavia, risulta essere molto restrittivo. Il suo ruolo educativo, infatti, può essere molto più significativo e potenzialmente molto più interessante. La robotica può dare un valido apporto a numerose discipline, fra cui fisica, l'elettronica, la biologia,

l'informatica e la psicologia. Ogni disciplina di questo elenco, facilmente estendibile, può avvantaggiarsi dell'uso di supporti robotici, specie se questi supporti siano liberamente configurabili, che non abbiano quindi delle forme immutabili ma si prestino a vari tipi di assemblaggi. Questi ultimi normalmente sono disponibili sottoforma di kit robotici (robot che l'utente deve montare da solo con l'aiuto di un manuale o un software) e sono presenti sul mercato in un buon numero. Il più ben riuscito, senza dubbio è il kit Mindstorms® NXT, prodotto e distribuito dall'azienda danese Lego®[2] per poche centinaia di euro. Sia la struttura, altamente configurabile (da sempre baluardo della filosofia dei mattoncini Lego), che la presenza di un set vario di sensori ne rende duttile l'utilizzo. Si possono, infatti, costruire robot di diverse forme: dai robot dotati di ruote a quelli dotati di gambe, avendo sempre la massima libertà di posizionare i sensori in qualunque parte del corpo del robot.

## *3. Robotics Attack!*

Art Attack[3] è un fortunato format televisivo distribuito in tutto il mondo e destinato ai bambini. Lo scopo della trasmissione è quello di insegnare ai bambini come realizzare facilmente delle piccole opere d'arte partendo da semplici strumenti quali carta, matite colorate, pennelli, materiali reperibili in casa e soprattutto tanta colla vinilica. Con Attacco di Robotica (Robotics Attack) intendiamo proporre questo approccio alla robotica educativa. Con pochi strumenti e un kit di robotica come quello Lego è possibile, infatti, cominciare ad esplorare il mondo della robotica a differenti livelli: dal più semplice di tutti, e anche quello meno interessante, che è costituito dalla robotica teleguidata a quello più ricco di stimoli rappresentato dalla robotica autonoma, ovvero quel campo della robotica che si occupa dei robot che non necessitano di alcun intervento umano. Nel nostro caso il collante, la nostra colla vinilica, sarà la Robotica Evolutiva [1], un campo di studio della ricerca robotica che trae ispirazione dalla teoria dell'evoluzione.

Noi abbiamo deciso di farci prendere da un attacco di robotica e di portare il risultato alla prossima edizione di Futuro Remoto che si terrà questo anno alla Città della Scienza di Napoli. Gli strumenti nel nostro caso sono semplici e facilmente reperibili: un kit Lego Mindstorms NXT, tre sensori infrarossi, una telecamera capace di estrarre blob di

[2] http://shop.lego.com
[3] http://:www.hitentertainment.com/artattack/

colore ed infine Evorobot*[4], un potente simulatore sviluppato al Laral (Laboratory of Autonomous Robotics and Artificial Life dell'Istituto di Scienze e Tecnologie della Cognizione del CNR, da Stefano Nolfi e Onofrio Gigliotta. Grazie ad Evorobot* non è richiesta, ai bambini o utenti aspiranti progettisti di robot, nessuna particolare conoscenza informatica, né tanto meno è necessaria nessuna competenza elettronica per costruirsi un robot con il kit Lego.


Figura 2. Kit Lego Nxt, NXTCam-v2 e sensore infrarosso DIST-Nx-Short-v2

## *4. Evorobot**

Evorobot* è un simulatore sviluppato principalmente per fare ricerca attraverso le tecniche della robotica evolutiva. Nella robotica evolutiva il controllore di un robot, in pratica l'oggetto che in base all'attivazione dei sensori deve decidere l'attivazione dei motori, è caratterizzato da un insieme di parametri, questi ultimi vengono posti in evoluzione proprio come in natura si evolvono gli esseri viventi. In Evorobot*, in particolare, il controllore è rappresentato da una rete neurale che imita, in qualche modo, il funzionamento del cervello. La rete, infatti, è composta da neuroni sensoriali (input), neuroni interni e neuroni motori (gli output) collegati tra loro da sinapsi il cui valore, detto peso sinaptico, rappresenta la forza della connessione tra unità neuronali.

Il processo evolutivo parte dalla produzione di una popolazione (per esempio di 100) casuale di insiemi di parametri (nel nostro caso pesi sinaptici), ogni insieme quindi viene testato, ovvero messo dentro l'oggetto controllore, per un certo periodo di tempo (misurato in time steps) definito tempo di vita. Ogni time step prevede un ciclo di in-

[4] http://laral.istc.cnr.it/evorobotstar

put/output del robot dove la relazione tra input (per es. sensori infrarossi) ed output (per es. motori) dipenderà dal particolare insieme di parametri usato. Le valutazioni del comportamento del robot vengono fatte attraverso una funzione di fitness che ci dice quanto è efficace il robot in un determinato comportamento. Una volta valutati tutti gli insiemi di parametri, che possiamo definire individui, questi vengono ordinati in base al loro punteggio, ovvero la loro fitness. I primi n individui (per esempio 20) vengono allora selezionati e riprodotti in modo da riformare la popolazione iniziale. La riproduzione, proprio come avviene in natura, è caratterizzata da mutazioni in modo da permettere l'emergenza di individui più adatti al particolare compito scelto, come per esempio quello dell'evitamento di ostacoli o di fuga da un predatore colorato percepibile dalla camera. Per fare evolvere un comportamento quindi è sufficiente una conoscenza informatica di base. L'utente deve soltanto riportare in un apposito file di configurazione le caratteristiche del robot che si vuole simulare (al momento sono disponibili Khepera, E-puck e Lego NXT) e dopodichè scegliere quale funzione di fitness utilizzare. Un utente esperto, comunque, ha sempre la possibilità di scrivere una sua particolare funzione di fitness, ma per il resto degli utenti basterà utilizzare le funzioni di fitness già presenti nel simulatore. Combinandole insieme, infatti, si possono ottenere una serie molto vasta di comportamenti. In alternativa l'utente potrà sempre utilizzare una forma di selezione artificiale come avviene in BreedBot[2], ovvero un processo di selezione diretta basata sull'osservazione degli individui.

## *5. BBot, come assemblare un robot in pochi passi*

Costruire un robot nell'immaginario collettivo richiede molte competenze, molto tempo e soprattutto molti soldi. La realtà, fortunatamente, è molto più semplice. Abbiamo cominciato col costruire una base per un robot dotato di due ruote (noi abbiamo progettato una base nuova per motivi di spazio, ma sul manuale Lego una vale l'altra). Successivamente abbiamo posto sul rostro tre sensori infrarossi (DIST-Nx-Short-v2 con un range che va da 4 a 40 cm) e una camera(NXTCam-v2)[5]. I sensori infrarossi, utili per percepire gli ostacoli, sono stai posti in modo da coprire un angolo di 90 gradi sul rostro del robot mentre la camera, capace di estrarre fino a otto blob di colore, è stata posta appena sopra il sensore infrarosso centrale. Per conferire a tutto una forma

[5] Reperibili sul sito http://www.mindsensors.com

più robusta(unico punto debole degli attuali kit robotici), infine, abbiamo aggiunto una scocca di vetroresina (questa ci serve solo a garantire l'integrità della forma durante un uso intensivo del robot. Per un uso normale non è affatto necessaria).

Il robot è pronto, e adesso? Adesso è il momento di farlo muovere, fargli fare qualcosa di interessante, di divertente e di educativo ovviamente. Per fare tutto ciò faremo uso della robotica evolutiva [1].

La robotica evolutiva applica gli strumenti dell'evoluzione alla robotica. Come ogni essere vivente, anche il nostro BBot, avrà un suo particolare codice genetico che codificherà indirettamente (attraverso una rete neurale) il suo repertorio comportamentale. Per testare i vari codici genetici ci avvaleremo del software Evorobot*. Attraverso Evorobot*, infatti, è possibile simulare facilmente il nostro robot Lego (basta selezionare la corretta posizione della camera e degli infrarossi e il gioco è fatto). A questo punto sarà l'utente, attivando le varie funzioni di fitness, a selezionare i robot simulati più interessanti. Le funzioni di fitness servono come valutatori per attivare il processo evolutivo. I robot, infatti, vengono valutati in base a quanto sono bravi a mostrare un dato comportamento, per esempio evitare gli ostacoli, andare verso il blue o scappare dal rosso etc.

Per il nostro BBot abbiamo scelto la funzione di fitness *"cattura il rosso*". Gli individui vengono premiati per la loro capacità di raggiungere un target di colore rosso. Nel giro di poche generazioni, circa 20 impiegando 1 minuto di un normale computer da tavolo, il nostro robot è pronto. A questo punto basta semplicemente scaricare il BBtot virtuale sul BBot reale e il gioco è fatto.

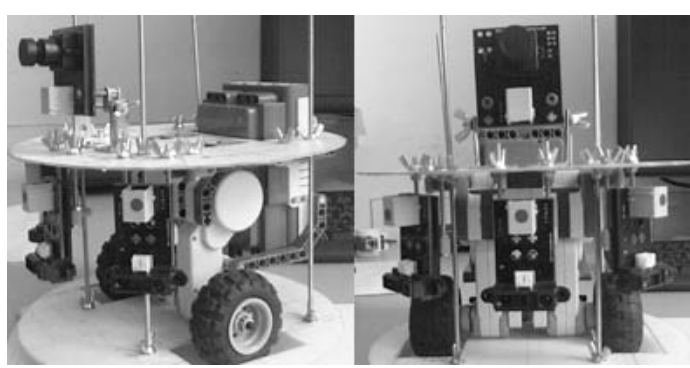


Figura 3. BBot in fase di assemblaggio

BBot a questo punto è pronto a spiegarci come funziona l'evoluzione e anche come può funzionare un cervello, visto che il suo sistema di controllo è una rete neurale, coniugando in modo semplice il versante educativo a quello di intrattenimento della robotica.

## *6. Conclusioni*

La robotica di intrattenimento prima e quella educativa dopo, sono entrate negli ultimi anni a far parte del nostro mondo. Ancora in punta di piedi stanno sugli scaffali dei giocattoli o nei negozi di elettronica, ma sicuramente nei prossimi anni avranno un maggiore impatto nelle nostre vite. La robotica evolutiva, unitamente all'uso di kit robotici liberamente assemblabili, renderà più accessibile il mondo robotico ai non professionisti del settore, portando la robotica autonoma ad una ampia diffusione. I robot come Robosapien o Pleo saranno ricordati come semplici pupazzi animati. Grazie agli sviluppi della robotica interna [3], inoltre, sarà anche possibile, permettere ai robot di mostrare stati emotivi che non siano semplici espressioni pre-programmate ma strettamente legate all'interazione controllore-robot-ambiente. Questo che stiamo vivendo, per concludere, si appresta ad essere il secolo della robotica [4] e la robotica evolutiva potrà essere una valida interfaccia per questa era tecnologica, permettendo a tutti di avere degli Attacchi di Robotica divertenti ed educativi.

## *Bibliografia*

1. Nolfi, S., Floreano, D.: Evolutionary Robotics. The Mit Press, Cambridge (2000)
2. Miglino, O., Gigliotta, O., Ponticorvo, M., Nolfi, S.: Breedbot: an evolutionary robotics application in digital content. The Electronic Library 26, 363-373 (2008)
3. Parisi, D.: Internal Robotics. Connection Scirnce, 16, 325-338 (2004)
4. Gates, B.: A Robot in Every Home, Scientific American Magazine, January (2007)

# *Conoscere la geometria senza rappresentazioni. Indicazioni da uno studio su organismi artificiali*

**Orazio Miglino**
*Dipartimento di Scienze Relazionali Università di Napoli "Federico II"*;
Via Porta di Massa, 1 80113 Napoli,
Tel. +39 0812535466, Fax +39 0812535634
orazio.miglino@unina.it

**Michela Ponticorvo**
*Dipartimento di Scienze Relazionali Università di Napoli "Federico II"*;
Via Porta di Massa, 1 80113 Napoli,
Tel. +39 0812535466, Fax +39 0812535634
michela.ponticorvo@unina.it

## *1. Introduzione*

La forma dell'ambiente è un'informazione comunemente utilizzata dagli animali per ri-orientarsi. In effetti molti vertebrati sfruttano questa informazione: gli scimpanzé, i piccioni, i ratti, gli esseri umani e i pulcini (Cheng e Newcombe, 2005) sono in grado di riconoscere le relazioni geometriche tra le mura di una stanza e usano quest'informazione per orientarsi in maniera efficace. Molte specie, insieme all' homo sapiens, hanno una comprensione chiara della geometria della porzione del mondo in cui vivono.

Un setting sperimentale molto diffuso per studiare l'utilizzo della geometria nel processo di ri-orientamento è l'open field box. Questo setting consiste di un'arena rettangolare nella quale il soggetto sperimentale deve orientarsi e localizzare una certa porzione dell'ambiente.

Al soggetto sperimentale viene mostrata la posizione di qualcosa di desiderabile, come del cibo o una via d'accesso ad un'area più interessante dell'ambiente. Dopo una procedura di disorientamento il soggetto viene riposizionato nell' arena e deve identificare la posizione del premio, generalmente in uno degli angoli. Basandosi sulla geometria il soggetto può identificare l'angolo corretto o l'angolo rotazionalmente equivalente (cioè l'angolo corrispondente all'angolo corretto dopo una rotazione di 180°). In questa condizione sperimentale gli animali com-

mettono sistematicamente questo tipo di errore detto errore rotazionale: in circa metà delle prove scelgono la posizione corretta, nell'altra metà scelgono la posizione rotazionalmente equivalente. Questo comportamento è stato interpretato come la prova che il cervello dei vertebrati possiede un modulo geometrico che codifica le informazioni geometriche dell'ambiente come la distanza e la direzione. Ulteriori esperimenti hanno mostrato che i ratti non solo usano la geometria, ma usano solo la geometria: utilizzano esclusivamente quest'informazione per orientarsi ignorando altre caratteristiche.

Partendo da questi data Gallistel (1990) postulò l'esistenza nel cervello dei ratti di un modulo neuro-cognitivo dedicato alla rappresentazione euclidea dello spazio di vita.

Il modulo geometrico ha ricevuto notevole attenzione durante gli ultimi 20 anni, ma recentemente questa ipotesi ha mostrato le prime crepe (Cheng, 2008). Se infatti appena qualche anno fa Cheng and Newcombe (2005) nella loro review sul tema avevano scritto della "natura obbligatoria dell'uso dell'informazione geometrica" gli stessi autori ora dubitano che questo sia vero. In alcuni casi degli indizi non-geometrici salienti possono oscurare l'apprendimento delle informazioni geometriche. La cosa interessante è che, anche se l'idea del modulo geometrico viene attualmente messa in dubbio, le ipotesi esplicative alternative si basano comunque sull'idea di rappresentazione (cfr. Cheng, 2008).

In questo contributo cercheremo di sostenere un differente punto di vista secondo il quale gli animali non usano una rappresentazione completa ed esplicita dello spazio, ma piuttosto raccolgono indizi significativi man mano che si muovono nell'ambiente, estraendo informazioni geometriche significative dalla costellazione di stimoli che ricevono dall'ambiente.

Secondo questo punto di vista, che ha solide radici nelle concezioni di filosofi come Dewey (1938) e Merleau-Ponty (1945), di psicologi come Piaget (1971), percettologisti come Gibson (1979) e più recentemente di O'Regan e colleghi (2001), di Clark (1997), del biologo ed epistemologo Varela (Maturana and Varela, 1980) e della scuola italiana di Parisi (Parisi et al., 1992; 1994 giusto per citarne alcuni), la conoscenza emerge dall'interazione tra un organismo, l'ambiente in cui vive e i vincoli fisici che esso pone. In altre parole la cognizione dello spazio sarebbe "situata".

Se si potesse entrare nella testa di questi animali, Gallistel si aspetterebbe di trovare una fotografia dell'ambiente, mentre i sostenitore dell'approccio "situato" cercherebbero una massa di dati indistinti che

diventano conoscenza solo quando l'animale usa queste informazioni durante l'azione.

Descriviamo adesso alcuni esperimenti condotti con organismi artificiali nei quali si evidenzia come l'abilità a sfruttare le informazioni geometriche non necessariamente si basa su una conoscenza/rappresentazione esplicita della geometria ambientale.

## *2. Materiali e Metodo*

### *2.1.L'organismo artificiale*

Il soggetto dei nostri esperimenti è Epuck, un mini-robot circolare (Mondada e altri, 2009), equipaggiato da un sistema sensoriale distale, uno prossimale e uno propriocettivo.

L'architettura neurale del sistema di controllo del robot è costituita da uno strato di neuroni di ingresso e uno strato di neuroni di uscita. Lo strato di ingresso è costituito da 8 neuroni sensoriali prossimali, 8 neuroni sensoriali distali e 2 neuroni sensoriali propriocettivi. Lo strato di uscita è formato da due neuroni motori e da un neurone "decisionale", definito di "localizzazione" che inibisce l'azione dei motori se supera una data soglia e ferma il robot per un certo lasso di tempo. Si interpreta l'azione del neurone di localizzazione come un atto di riconoscimento differenziato delle diverse zone di un ambiente.

### *2.2. L'ambiente*

Nei nostri esperimenti abbiamo ripreso il setting sperimentale dell'open field box descritto nell'introduzione. L'ambiente di addestramento è costituito da un'arena rettangolare circoscritta da pareti bianche. Considerate le caratteristiche della telecamera di bordo, gli incroci angolari sono stati colorati di nero per renderli percepibili dal robot. In corrispondenza di un angolo è stata definita una zona di rinforzo (vedi sotto).

### *2.3. Procedura di evoluzione*

I robot sono stati evoluti con un Algoritmo Genetico. Ogni volta che il robot raggiunge ed identifica il target (ovvero l'attivazione del

neurone di localizzazione supera una data soglia), ottiene in premio un punto di "fitness". Ciascun robot viene testato 4 volte e riceve un punteggio finale di fitness equivalente alla somma dei punti ricevuti durante tutte le prove. Alla fine di questa procedura, gli 80 robot con i punteggi di fitness più bassi vengono eliminati (selezione per troncamento). I restanti 20 invece sono clonati con una percentuale di mutazione del 3%. Il ciclo di test, selezione e clonazione è stato ripetuto per 100 "generazioni".

Al fine di investigare se e come gli organismi artificiali sfruttano le diverse tipologie di informazione (distale, prossimale, propriocettiva) per raggiungere la conoscenza dell'ambiente in cui sono immersi e, quindi, produrre un efficiente comportamento di orientamento spaziale abbiamo effettuato quattro esperimenti. Nel primo esperimento abbiamo usato dei robot equipaggiati con tutti i loro apparati sensoriali (telecamera, sensori all'infrarosso, registrazione dell'attivazione dei motori); nel secondo esperimento abbiamo concesso ai robot il solo uso della telecamera (stimolazione distale); il terzo esperimento è stato condotto con robot stimolati esclusivamente dai sensori all'infrarosso (stimolazione prossimale) ed infine, il quarto esperimento è stato effettuato su popolazioni di robot essenzialmente "ciechi", che avevano cioè a disposizione solo le informazioni di tipo propriocettivo (feedback motorio).

## *3. Risultati*

Nella figura seguente sono riportati gli indici comportamentali relativi ai 4 esperimenti confrontati con i dati relativi all'esperimento sui pesci come esempio di un organismo naturale.

La figura 1 riporta gli indici comportamentali dei migliori organismi dell'ultima generazione. Inoltre, vengono riportati i dati registrati da Sovrano e altri (2002) negli esperimenti con i pesci, per facilitare il confronto naturale/artificiale.

Le percentuali di scelta riportate mostrano che, come accade nel caso dei vertebrati, gli organismi artificiali scelgono con frequenza simile l'angolo corretto e quello rotazionalmente equivalente. Gli organismi dotati di un apparato sensoriale completo, così come quelli dotati di telecamera effettuano una scelta corretta o commettono un errore rotazionale nella totalità dei casi, mentre per gli organismi che possono basarsi solo sul feedback motorio (stimolazione propriocettiva) o sull'informazione prossimale, questa percentuale scende, anche se in misura non rilevante.

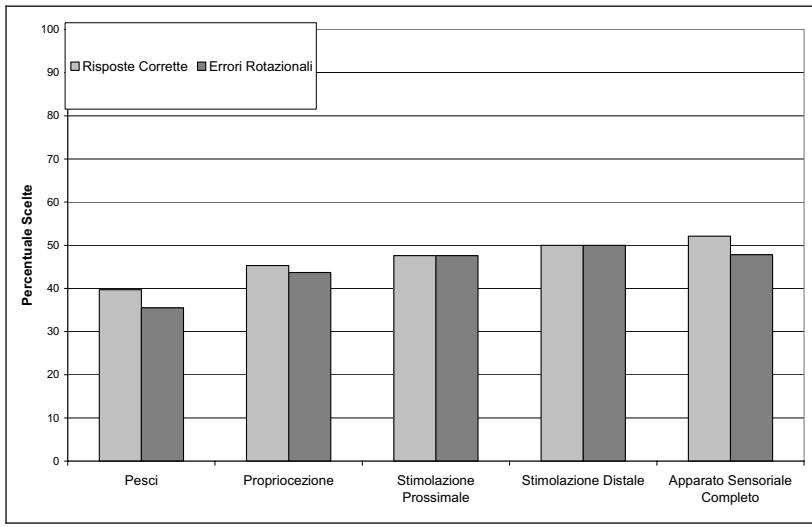



Figura 1. Percentuale di risposte corrette ed errori rotazionali nei pesci (dati da Sovrano e altri (2002)) e per gli organismi artificiali. La percentuale delle altre scelte non è riportata.

## *4. Discussione*

Dai dati sopra riportati si può dedurre che sebbene vi siano delle differenze quantitative, tutte le diverse tipologie sensoriali sono sufficienti a sollecitare negli organismi artificiali un comportamento simile a quelli dei pesci. In sostanza, in tutte le condizioni sperimentali, i robot si comportano "come se" conoscessero la forma dell'ambiente;

Questi risultati mostrano che, pur con le differenze dovute alla quantità di informazioni disponibile per il robot, non è necessaria una rappresentazione esplicita della geometria dell'ambiente per svolgere correttamente il compito di localizzazione (cfr. Ponticorvo e Miglino 2009; Miglino, Ponticorvo e Bartolomeo, 2009). Al contrario questi dati sono perfettamente compatibili con un'interpretazione del comportamento in termini di strategie senso-motorie. In sostanza l'interpretazione in termini di cognizione situata è valida tanto quella rappresentazionalista, che chiama in causa il modulo geometrico e, per molti versi, è ad essa preferibile perché implica un numero minore di variabili e dei meccanismi più semplici.

## *Bibliografia*


1. Cheng, K. (2008). Whither geometry? Troubles of the geometric module, Trends Cognitive Science 12, 355–361
2. Cheng, K., Newcombe, N.S. (2005). Is there a geometric module for spatial orientation? Squaring theory and evidence. Psychonomic Bulletin and Review, 12, 1-23.
3. Clark, A. (1997). Being There: Putting Brain, Body, and World Together Again. Cambridge, Massachussets: MIT Press.
4. Dewey, J. (1938). Experience and Education, New York: Macmillan
5. Gallistel, C. R. (1990). The Organization of Learning. Cambridge MA: MIT Press.
6. Gibson, J.J. (1979). The Ecological Approach to Visual Perception. Boston: Houghton Mifflin
7. Maturana, H.R., Varela, F.J.(1980). Autopoiesis and Cognition. Reidel, Boston
8. Merleau-Ponty, M. (1945), Phenomenologie de la perception, Gallimard, Paris.
9. Miglino, O., Ponticorvo, M., Bartolomeo, P. (2009). Place cognition and active perception. A study with evolved robots. Connection Science, 21-1, 3-14.
10. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D. and Martinoli, A. (2009) The e-puck, a Robot Designed for Education in Engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, 1(1) pp. 59-65.
11. O'Regan, J.K., Noë A. (2001). A sensorimotor account of vision and visual consciousness. Behavior and Brain Sciences, 24 (5).
12. Parisi, D. (1994). Are neural networks necessarily passive receivers of input? In Masulli, F., Morasso, P.G.and Schenone, A. (eds.).Neural networks in biomedicine. Singapore, World Scientific, 113-124
13. Parisi, D., Nolfi, S., Cecconi, F. (1992). Learning, Behavior, and Evolution in Varela, F, Bourgine, P. (eds.). Toward a practice of autonomous systems. MIT Press, 207-216
14. Piaget, J. (1971). Biology and Knowledge: An Essay on the Relations between Organic Regulations and Cognitive Processes. Chicago University Press.
15. Ponticorvo, M., Miglino, O. (2009). Encoding geometric and non-geometric information: a study with evolved agents. Animal Cognition, DOI 10.1007/s10071-009-0255-7.

16. Sovrano, V. A., Bisazza, A., & Vallortigara, G. (2002). Modularity and spatial reorientation in a simple mind: Encoding of geometric and non-geometric properties of spatial environment by fish. Cognition, 85, 51-59.

# *Apprendimento cumulativo basato su motivazioni intrinseche: un modello neurale gerarchico testato su un robot simulato*

**Marco Mirolli**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia 44, 00185 Roma
Tel. +39 06 44595231, Fax +39 06 44595243
marco.mirolli@istc.cnr.it

**Massimiliano Schembri**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia 44, 00185 Roma
Tel. +39 06 44595231, Fax +39 06 44595243
massimiliano.schembri@istc.cnr.it

**Gianluca Baldassarre**
*Istituto di Scienze e Tecnologie della Cognizione, CNR*
Via San Martino della Battaglia 44, 00185 Roma
Tel. +39 06 44595231, Fax +39 06 44595243
gianluca.baldassarre@istc.cnr.it

## *1. Introduzione*

I robot di oggi sono molto limitati rispetto al numero di compiti che possono svolgere ad al grado di flessibilità che hanno nello svolgere tali compiti. In effetti, la stragrande maggioranza dei robot oggi è programmata (o evoluta) per compiere un singolo compito in un solo tipo di ambiente. Gli organismi naturali, e gli esseri umani in particolare, sono invece in grado di fare molte cose diverse e di adattare le loro capacità alle nuove sfide che l'ambiente pone loro. Ciò è possibile grazie alla loro capacità di apprendere un gran numero di abilità in modo cumulativo sulla base di motivazioni intrinseche come la curiosità e la novità [1-2], e di risolvere i problemi che si trovano ad affrontare combinando e ricombinando in modo flessibile le abilità apprese. Lo studio di sistemi artificiali in grado di modellare e riprodurre le capacità di ap-

prendimento autonomo basato su motivazioni intrinseche degli organismi reali è un campo in rapida espansione negli ambiti della machine learning, della robotica autonoma e della vita artificiale [3-5].

In questo contributo presentiamo un modello gerarchico di apprendimento per rinforzo basato su motivazioni intrinseche che è in grado di far apprendere ad un robot autonomo (simulato) una serie di abilità che sono successivamente combinate per risolvere differenti compiti sulla base di rinforzi esterni.

## *2. Modello*

### *2.1 Robot e ambiente*

Il robot simulato è un robot mobile di 30 centimetri di diametro munito di una video camera puntata su una porzione di terreno di fronte al robot di 24x8 cm. L'input del robot consiste in 12x3 valori binari che corrispondono all'attivazione di recettori RGB (rosso-verde-blu) che fanno il campionamento dell'immagine della videocamera divisa in una griglia regolare di 2x6 (fig. 1). L'output motorio del robot consiste di due unità che regolano rispettivamente la variazione di orientazione (in un intervallo di [-30, 30] gradi), e la velocità di traslazione (in un intervallo di [0, 2] cm/sec. L'ambiente consiste in un'arena quadrata con un pavimento colorato in modo regolare (fig. 2).

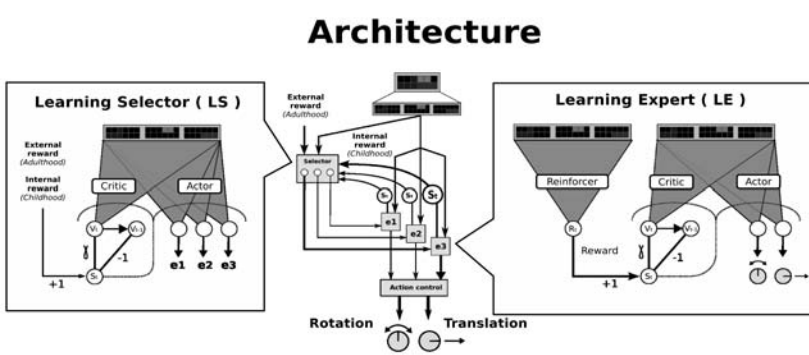



Figura 1. Il sistema di controllo

### *2.2 Vita e compito*

La 'vita' del robot è divisa in due fasi: infanzia ed età adulta. Durante l'infanzia il robot è libero di esplorare l'ambiente apprendendo un insieme di abilità sensomotorie sulla base di motivazioni interne (senza alcun rinforzo esterno). Durante la vita adulta il robot apprende a combinare le abilità acquisite per risolvere sei compiti differenti sulla base di rinforzi esterni. I sei compiti sono raffigurati in figura 2. Ciascuno di essi corrisponde ad una differente fase adulta ed inizia con il robot posizionato in una particolare posizione iniziale: l'obiettivo del robot è raggiungere il maggior numero di volte possibile il rinforzo, la cui locazione nell'ambiente dipende a sua volta dal compito. Ogni volta che il robot raggiunge il rinforzo viene riposizionato nella posizione iniziale.

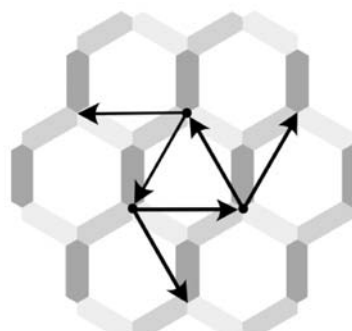


Figura 2. L'ambiente. Le strisce sono colorate di blu (grigio scuro), rosso (grigio) e verde (grigio chiaro). Lo sfondo, qui rappresentato come bianco, è in realtà nero. Ciascuna freccia rappresenta uno dei sei compiti del robot. L'inizio della freccia rappresenta la posizione di partenza mentre la punta rappresenta il punto in cui è presente il rinforzo.

### *2.3 Il sistema di controllo*

Il sistema di controllo del robot consiste in una rete neurale modulare organizzata in modo gerarchico, e formata da un 'selettore' e da un certo numero di 'esperti', in questo caso 3 (figura 1). Ciascun esperto rappresenta quella che abbiamo chiamato 'abilità', ossia una regola di mapping sensomotorio tra la percezione corrente e il movimento da

eseguire. Il selettore decide invece, sulla base dell'input, quale degli esperti assume in ciascun istante il controllo dei movimenti del robot.

Sia il selettore che ciascuno degli esperti consiste in una implementazione neurale del modello attore-critico dell'apprendimento per rinforzo [6]. Tale modello è composto da un attore, che mappa gli input percettivi negli output (che rappresentano le azioni del modello) ed un critico, che mappa gli input in una valutazione dello stato percettivo corrente, che, con il proseguire dell'apprendimento, tenderà a rappresentare la somma di tutti i rinforzi futuri scontata rispetto al ritardo temporale con cui tali rinforzi sono attesi. Sia l'attore che il critico apprendono sulla base della 'sorpresa', che è una misura dell'errore di predizione del rinforzo del critico.

Ciascun esperto comprende anche un'ulteriore rete neurale, chiamata 'rinforzatore', che mappa l'input corrente nel segnale di rinforzo che guida l'apprendimento di quell'esperto durante l'infanzia del robot. Il segnale di rinforzo per il selettore durante l'infanzia è invece costituito dalla sorpresa (definita) sopra dell'esperto a cui il selettore ha dato il controllo nel ciclo precedente. Dato che la sorpresa può essere considerata come un'indice di quando un esperto sta apprendendo in ciascun istante, durante la fase esplorativa il selettore impara così a dare il controllo a quegli esperti che stanno imparando meglio.

Durante la fase adulta invece l'apprendimento degli esperti è interrotto, mentre il selettore impara a combinare le abilità acquisite dagli esperti sulla base del rinforzo esterno.

### *2.4 L'algoritmo genetico*

I pesi delle reti neurali che costituiscono attori e critici del selettore e degli esperti vengono dunque appresi durante la vita del robot. I pesi dei rinforzatori di ciascun esperto sono invece fissi, e sono evolute tramite un algoritmo genetico, secondo il tipico schema della robotica evolutiva [7] L'insieme dei pesi dei rinforzatori costituisce infatti il genoma di un individuo. All'inizio di una simulazione 100 individui vengono inizializzati con pesi casuali e vengono fatti vivere nel loro mondo per le due fasi dell'infanzia e dell'età adulta. La fitness di ciascun individuo è calcolata solo durante gli ultimi 50000 passi della vita adulta per ciascuno dei 6 compiti, e viene misurata come il numero di rinforzi raggiunti normalizzato per il valore massimo teoricamente possibile, ossia quello che sarebbe raggiunto da un individuo che andasse sempre in linea retta alla massima velocità dal punto di partenza al punto dove è il rinforzo.

La generazione successiva viene prodotta copiando 5 volte il genoma di ciascuno dei 20 migliori individui ed aggiungendo mutazioni casuali nei pesi. L'intera procedura è ripetuta per 100 generazioni.

### *3. Risultati*

In linea generale, i risultati sono abbastanza promettenti: sia la fitness media che quella del miglior individuo di una generazione crescono molto velocemente e raggiungono un valore stabile in poche generazioni (tipicamente 10-20). La fitness del miglior individuo supera 0.8, che è un valore altissimo considerato che una fitness uguale ad 1 indicherebbe che il robot va sempre in linea retta alla massima velocità verso il rinforzo (ossia seguendo le frecce di figura 4), il che è impossibile dato che il robot non può vedere il rinforzo e può far affidamento solo sull'informazione locale data dal colore del pavimento di fronte a sé.

La figura 3 mostra il comportamento del migliore robot evoluto. Fig. 3a mostra il comportamento all'inizio dell'infanzia: dato che all'inizio i pesi di tutti gli attori sono casuali, il comportamento non può che essere casuale. In particolare, il selettore assegna in modo casuale il controllo ai vari esperti, che a loro volta agiscono a caso. Alla fine dell'infanzia però il robot mostra un comportamento altamente strutturato (fig. 3b): in particolare, ciascun esperto si è specializzato nel seguire una linea colorata (apprendendo sulla base del proprio rinforzatore evoluto), mentre il selettore, che ha appreso sulla base della sorpresa degli esperti (che a sua volta è una misura del grado di apprendimento), ha appreso ad assegnare il controllo agli esperti che sono stati in grado di massimizzare il proprio apprendimento.

Il risultato di questo processo di sviluppo è che alla fine dell'infanzia il robot ha acquisito un insieme di abilità di base che possono essere usate per risolvere i vari compi incontrati durante la vita adulta. Questo è illustrato chiaramente in fig. 3c-h, che mostrano il comportamento del robot adulto alla fine della fase adulta per ciascuno dei 6 compiti. Ogniqualvolta il robot si trova su una delle strisce colorate il selettore dà il controllo all'esperto specializzato per seguire quel colore (a parte rari casi dovuti alla natura parzialmente stocastica della selezione). Quando una striscia colorata finisce ed il robot arriva ad un incrocio, il selettore deve solo apprendere a selezionare l'esperto che è in grado di seguire il colore che porta al rinforzo. Il risultato è che il robot è in grado di apprende velocemente a risolvere i diversi compiti che deve affrontare.

Ciò è possibile grazie al fatto che il robot non ha bisogno di imparare tutto da capo per risolvere ciascun singolo compito, ma può farlo semplicemente combinando appropriatamente le abilità acquisite durante l'infanzia.

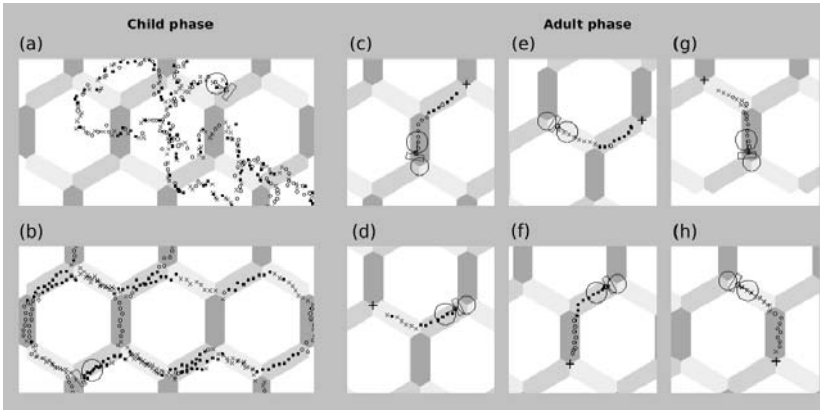



Figura 3: Rappresentazioni del comportamento del miglior robot evoluto in varie fasi. Il robot è rappresentato come un cerchio con un rettangolo davanti (l'area percepita tramite la telecamera). I piccoli simboli (quadrati neri, cerchi vuoi e croci) indicano quale dei tre esperti è stato selezionato in una data posizione. (a) Inizio dell'infanzia. (b) Fine dell'infanzia. (c-h) La fine dell'ultima fase della vita adulta per ciascuno dei 6 compiti. Le croci ed I cerchi agli incroci delle strisce colorate rappresentano, rispettivamente, le posizioni iniziali e finale.

## *4. Conclusione*

In questo contributo abbiamo presentato un'architettura neurale modulare e gerarchica disegnata per rendere possibile l'apprendimento cumulativo basato su motivazioni intrinseche di sistemi autonomi. Abbiamo testato la nostra architettura con una simulazione in cui un robot simulato dotato di un sistema visivo molto semplice impara a risolvere diversi compiti di navigazione tramite la combinazione delle abilità apprese durante la sua fase di sviluppo sulla base di rinforzi interni provenienti da (a) rinforzatori evoluti tramite un algoritmo genetico e (b) la 'sorpresa', ossia l'errore nella predizione dei rinforzi futuri, che rappresenta una misura del tasso di apprendimento.

## *Riconoscimenti*

IM-CLeVeR è supportato dalla Commissione Europea nell'ambito dell'iniziativa 'FP7 - Cognitive Systems, Interaction, and Robotics', con il contratto numero 231722.

## *Bibliografia*

1. White, R.W.: Motivation Reconsidered: The Concept of Competence. Psychological Review, 66 (5), 297-333 (1959)
2. Berlyne, D.E.: Conflict, Arousal and Curiosity. McGraw-Hill, New York (1960)
3. Schmidhuber, J.: A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In J-A. Meyer, and S.W. Wilson, (Eds.), From Animals to Animats, The MIT Press, Cambridge, MA, pp. 222-227 (1991)
4. Barto, A.G., Singh, S., Chentanez, N.: Intrinsically Motivated Learning of Hierarchical Collections of Skills. Proceedings of the Third International Conference on Development and Learning (2004)
5. Oudeyer, P.Y., Kaplan, F., Hafner, V.: Intrinsic Motivation Systems for Autonomous Mental Development. IEEE Transactions on Evolutionary Computation, 11 (2), 265-286 (2007)
6. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge MA (1998)
7. Nolfi, S., Floreano, D.: Evolutionary robotics. The biology, intelligence, and technology of self-organizing machines. The MIT Press, Cambridge, MA. (2000)

# *Riduzione delle chiamate della funzione di fitness tramite approssimazione della fitness per algoritmi evolutivi*

**Fabio Moretti**
*Dipartimento di Informatica e Automazione (DIA), Università degli Studi "Roma Tre"*;
Via della Vasca Navale, 79 00146 Roma,
Tel. +39 0630484411, Fax +39 0630484811
moretti@dia.uniroma3.it

**Matteo De Felice**
*Centro Ricerche Casaccia, ENEA*;
Via Anguillarese, 301 00060 Roma,
Tel. +39 0630484411, Fax +39 0630484811
matteo.defelice@enea.it

## *1. Introduzione*

Metodologie di ottimizzazione sono utilizzate in ogni problema per indirizzare al meglio le fasi di un processo (progettazione, design o controllo, ad esempio).

Gli algoritmi evolutivi, tra cui gli algoritmi genetici (GA), sono algoritmi meta-euristici di ottimizzazione che si basano su meccanismi ispirati all'evoluzione e alla selezione naturale. Tali algoritmi cercano la soluzione ottima di un problema modellato tramite una funzione di fitness, che calcola la qualità di una soluzione. Il calcolo di tale funzione per alcuni problemi reali può richiedere un dispendio elevato a livello computazionale oppure economico, qualora per valutare una soluzione sia necessaria una spesa per il consumo di materiali o di energia elettrica, oppure può accadere che una forma analitica di tale funzione può non essere disponibile o molto rumorosa.

In tali casi può essere opportuno usare un'approssimazione della funzione di fitness. Le tecniche di Fitness Approximation (FA) consentono di diminuire il numero di richieste di calcolo della fitness utiliz-

zando valori stimati oltre ai valori ottenuti dalla funzione originaria. I primi studi e applicazioni al riguardo risalgono al 1985 (Grefenstette et al., 1985) mentre per gli sviluppi più recenti si vedano (Branke et al., 2005) e (Jin, 2005).

L'obiettivo dell'articolo è dimostrare come l'uso della Fitness Approximation applicata ad algoritmi genetici su alcune note funzioni di benchmark possa portare ad una diminuzione drastica sull'uso della funzione di fitness senza portare ad un decadimento evidente dei risultati.

Nel Capitolo 2 vengono presentati alcuni approcci sull'uso della FA. Il Capitolo 3 descrive la realizzazione, i test ed i risultati di un algoritmo di ottimizzazione supportato dalla FA. Il Capitolo 4 conclude l'articolo presentando alcune considerazioni finali.

## *2. Fitness Approximation*

I processi di ottimizzazione richiedono un numero spesso consistente di accessi alla funzione obiettivo, o più in generale al processo che si desidera ottimizzare. La FA consente di ridurre il numero di accessi effettuando una stima del valore della fitness anziché calcolare la fitness stessa. La maniera in cui la stima viene effettuata stabilisce la tipologia di FA utilizzata. Gli approcci più comuni per l'approssimazione fanno uso di polinomi (Carpenter, 1992), reti neurali (Carpenter, 1992, Hong, 2003), clustering (Kim, 2001) e modelli di Kriging (Willmes, 2003)

### *2.1 Approssimazione Polinomiale*

L'approssimazione polinomiale è uno degli approcci più semplici e più utilizzati. La forma più comune è la seguente:

$$y = \beta_0 + \sum_{i=1}^{K} \sum_{j=1}^{N} \beta_{ij} x_j^i$$

$\beta_{ij}$ sono i coefficienti del polinomio che approssima l'andamento della funzione obiettivo. L'approssimazione polinomiale necessita di un numero crescente di valori della funzione da approssimare al crescere della dimensionalità del problema per raggiungere un buon fitting.

### *2.2 Reti Neurali con Funzioni a Base Radiale*

Le Funzioni a Base Radiale (RBF) sono quelle funzioni il cui valore di uscita dipende unicamente dalla distanza tra il valore di ingresso ed un punto fissato chiamato centro della RBF. Le funzioni RBF più comuni sono:

- Gaussiana: $f(r) = \exp(-cr^2)$
- Multiquadratica: $f(r) = \sqrt{r^2 + c^2}$
- Multiquadratica inversa: $f(r) = \frac{1}{\sqrt{r^2 + c^2}}$
- Cauchy: $f(r) = -\frac{c^2}{r^2 + c^2}$

dove $r$ è la distanza dal centro e $c$ un parametro della funzione.

Una rete neurale RBF è costituita da uno strato di input, dipendente dalla dimensione del problema e uno strato nascosto, i cui neuroni non sono altro che funzioni a base radiale ed uno strato di output. I neuroni di ogni singolo strato, come nelle reti neurali più comuni, sono tutti collegati con archi pesati con i neuroni dello strato successivo. I pesi si possono trovare con diverse tecniche di ottimizzazione (addestramento). Per un'introduzione completa alle reti neurali RBF rimandiamo a (Haykin, 1998).

Rispetto all'approssimazione polinomiale una rete neurale è computazionalmente più complessa ed è in grado di modellare delle relazioni anche non-lineari tra ingressi ed uscite.

## *3. Applicazione della Fitness Approximation*

Per mostrare l'effettivo risparmio in termini di chiamate alla funzione obiettivo originaria con la FA è stato realizzato in MATLAB un algoritmo che effettua un'ottimizzazione supportata dalla Fitness Approximation. Lo schema generico di funzionamento è il seguente:

1. Si calcola l'approssimazione della fitness basandosi sui valori noti, ovvero in punti in cui già si è calcolata in precedenza la fitness reale
2. Si esegue l'algoritmo di ottimizzazione su tale approssimazione

3. La soluzione ottenuta al termine dell'ottimizzazione viene valutata usando la funzione di fitness reale
4. Se non si è raggiunto il criterio di stop si torna al passo 1.

All'inizio della procedura di ottimizzazione l'insieme dei punti "noti" viene riempito in maniera casuale, scegliendo un punto nello spazio delle soluzioni e valutandone la fitness.

### *3.1 Test e Risultati*

Nella realizzazione dei test sono state utilizzate come funzioni obiettivo le funzioni di Rastrigin, Ackley e Griewank.

Le caratteristiche del GA utilizzato applicate indistintamente su tutte le funzioni obiettivo sono le seguenti:

- Selezione: Roulette Wheel Selection
- Crossover: Single Point con probabilità 1
- Mutazione: Uniforme con probabilità 0.05
- Numero di individui iniziale: 5

Per ogni funzione gli individui della popolazione di partenza sono stati creati all'interno degli intervalli mostrati nella seguente Tabella 1.

Tabella 1: intervalli utilizzati per le funzioni di benchmark

| | Rastrigin | Ackley | Griewank |
|---|---|---|---|
| Intervallo | [-5.12, 5.12] | [-30, 30] | [-600, 600] |

Abbiamo confrontato l'algoritmo genetico (GA) con l'approccio basato su Fitness Approximation descritto nel Paragrafo 3 utilizzando una rete RBF come approssimazione, denominato GA-FA. Per un confronto più completo abbiamo utilizzato anche un algoritmo classico di Evolution Strategies (ES) del tipo (1+1)-ES con sigma adattivo tramite *fifth-rule* (Beyer, 2002).

Abbiamo eseguito 100 esperimenti per ogni funzione di benchmark e calcolato valori medi e deviazione standard. Abbiamo imposto come criterio di stop il raggiungimento di un numero determinato di chiamate alla funzione di fitness reale (denominata da noi f.r.).

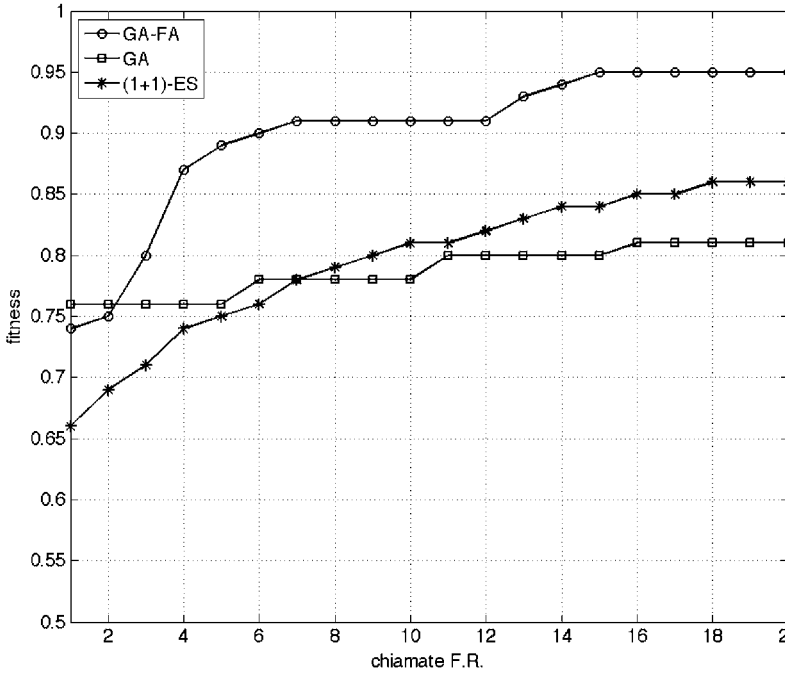



Figura 1. Confronto sulla funzione di Rastrigin

Nel grafico in figura 1 si confrontano gli algoritmi utilizzati sulla funzione di Rastrigin. La funzione di fitness è calcolata nel modo seguente:

$$f(x) = 1 - \frac{e_{min} - e_x}{e_{max}}$$

dove con $e_{min}$ e $e_{max}$ abbiamo i minimi e i massimi (noti a priori) delle funzioni considerate. Per motivi di chiarezza non abbiamo inserito nel grafico anche l'ampiezza della deviazione standard dei valori al termine dei 100 run, nella Tabella 1 sintetizziamo i risultati sulla funzione di Rastrigin.

Tabella 2: Risultati riassuntivi sulla funzione di Rastrigin

| N.chiamata F.R. | GA-FA | Dev. Std. GA-FA | GA | Dev. Std. GA | (1+1)-ES | Dev. Std. (1+1)-ES |
|---|---|---|---|---|---|---|
| 5 | 0.89 | 0.08 | 0.76 | 0.11 | 0.75 | 0.12 |
| 10 | 0.91 | 0.07 | 0.78 | 0.11 | 0.81 | 0.10 |
| 20 | 0.94 | 0.04 | 0.81 | 0.10 | 0.86 | 0.08 |

Tabella 3: Risultati riassuntivi sulla funzione di Griewank

| N.chiamata F.R. | GA-FA | Dev. Std. GA-FA | GA | Dev. Std. GA | (1+1)-ES | Dev. Std. (1+1)-ES |
|---|---|---|---|---|---|---|
| 5 | 0.90 | 0.07 | 0.89 | 0.07 | 0.82 | 0.16 |
| 10 | 0.94 | 0.05 | 0.89 | 0.07 | 0.90 | 0.13 |
| 20 | 0.96 | 0.03 | 0.91 | 0.08 | 0.96 | 0.10 |

Tabella 4:Risultati riassuntivi sulla funzione di Ackley

| N.chiamata F.R. | GA-FA | Dev. Std. GA-FA | GA | Dev. Std. GA | (1+1)-ES | Dev. Std. (1+1)-ES |
|---|---|---|---|---|---|---|
| 5 | 0.18 | 0.06 | 0.24 | 0.15 | 0.25 | 0.17 |
| 10 | 0.22 | 0.11 | 0.25 | 0.14 | 0.35 | 0.22 |
| 20 | 0.26 | 0.13 | 0.28 | 0.14 | 0.50 | 0.26 |

In figura 2 e 3 sono visibili gli stessi confronti sulla funzione di Griewank e di Ackley, rispettivamente. Nelle Tabelle 3 e 4 sono visibili i risultati con le deviazioni standard.

## *4. Discussione e conclusioni*

Nell'articolo abbiamo cercato di investigare l'utilità dell'uso della Fitness Approximation nel ridurre il numero di valutazioni della funzione di fitness. Nelle funzioni da noi selezionate per il confronto abbiamo osservato, nelle funzioni di Rastrigin e Griewank, in media un andamento migliore del GA-FA rispetto al GA e (1+1)-ES. La capacità

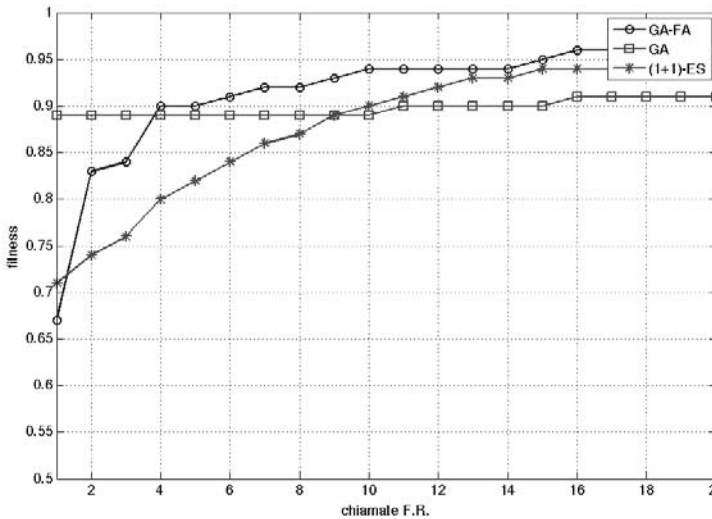



Figura 2. Confronto sulla funzione di Griewank

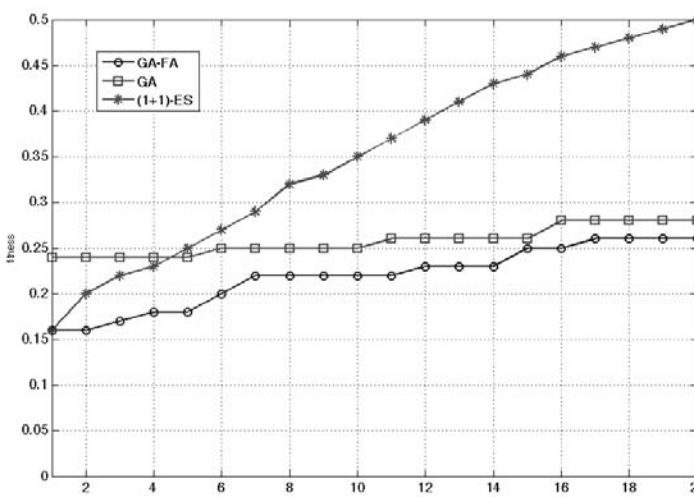



Figura 3. Confronto sulla funzione di Ackley

dell'approssimazione di portare le soluzioni ad esplorare parti dello spazio con fitness più alta cresce come è ovvio all'aumentare delle informazioni di cui si dispone, quindi dei valori di fitness reali. La differenza di prestazioni tra le diverse funzioni da noi studiate può essere spiegata da un differente *fitness landscape* che può risultare più difficile da approssimare per certe funzioni. L'approssimazione quindi non riesce ad adattarsi ai numerosissimi minimi locali ed all'andamento piuttosto irregolare della funzione.

Osserviamo comunque da questi esperimenti come l'incremento di velocità nella convergenza della fitness, come già evidenziato in (Branke et al., 2005), possa essere una buona motivazione per utilizzare tecniche di FA di supporto all'ottimizzazione tramite algoritmi evolutivi.

Pensiamo che i vantaggi della Fitness Approximation siano più evidenti nei problemi dove la complessità non risieda nella multimodalità e irregolarità della funzione, fenomeni volutamente esasperati nelle funzioni di benchmark e meno frequentemente osservabili nei problemi reali. Infatti la necessità di ridurre il numero di valutazioni della fitness è particolarmente sentita nell'ottimizzazione di processi reali dove la valutazione della bontà di una soluzione è costosa o particolarmente complessa da ottenere ed è proprio su problemi di tale genere verteranno i nostri lavori futuri.

## *Bibliografia*

1. Jin, Y. A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 3-12 (2005)
2. Haykin, S. Neural Networks: A Comprehensive Foundation (2nd Edition*)* (Prentice Hall, 1998), 2 edn.
3. Beyer, H.-G. & Schwefel, H.-P. Evolution strategies – a comprehensive introduction. Natural Computing, 3-52 (2002)
4. Grefenstette, J. J. & Fitzpatrick, J. M. Genetic search with approximate function evaluation. In Proceedings of the 1st International Conference on Genetic Algorithms, 112-120 (L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985)
5. J. Branke e C. Schmidt. Fast convergence by means of fitness estimation. In: Soft Computing Journal, pp 13-20 (2005)
6. Y.S. Hong, H. Lee, M.J. Tahk. Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural network. In: Engineering Optimization, pp 91-102 (2003).

7. W. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. Technical report 92-2247, AIAA, 1992
8. H.S. Kim e S.B. Cho. An efficent Genetic Algorithm with less fitness evaluation by clustering. In: Proceeding of the Congress on Evolutionary Computation, pp 887-894 (2001)
9. L. Willmes, T. Baeck, Y. Jin, and B. Sendhoff. Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In Proceedings of IEEE Congress on Evolutionary Computation, pages 663-670, 2003

# *Valutazione dell'effetto della sincronizzazione nella Particle Swarm Optimization*

**Luca Mussi**
*Dipartimento di Ingegneria dell'Informazione, Università di Parma*;
Viale G. Usberti, 181A - I 43100 Parma
mussi@ce.unipr.it

**Stefano Cagnoni**
*Dipartimento di Ingegneria dell'Informazione, Università di Parma*;
Viale G. Usberti, 181A - I 43100 Parma
cagnoni@ce.unipr.it

**Fabio Daolio**
*Informatica Systems Institute (ISI)*
*HEC, Università di Losanna, Svizzera*
fabio.daolio@unil.ch

*Nonostante la notevole popolarità della Particle Swarm Optimization (PSO), i meccanismi che ne regolano il comportamento sono ancora oggetto di ricerca. Riguardo la comunicazione tra particelle, molti autori discutono gli effetti della topologia dello sciame, ma pochi la dinamica dello scambio di informazioni. In questo lavoro mostriamo che l'aggiornamento sincrono degli attrattori sociali influenza l'efficacia della PSO, confrontandone varianti sincrone ed asincrone su un benchmark. I risultati mostrano che la topologia "global best" è sensibile alla politica di aggiornamento, soprattutto per elevate dimensioni dello spazio di ricerca. Al contrario, topologie poco connesse non sembrano risentire della sincronizzazione.*

## *1. Introduzione*

La Particle Swarm Optimization (PSO) fu introdotta da Kennedy e Eberhart nel 1995 [11]. La PSO ricerca l'ottimo di una funzione (fitness) imitando il movimento degli stormi di uccelli in cerca di cibo. Le particelle "volano" sul dominio della funzione di fitness di cui, ad ogni passo, sono valutati i valori associati alle nuove posizioni. Ogni particella conserva parte della propria velocità (inerzia), essendo inoltre soggetta a due forze attrattive: l'attrazione cognitiva la attira verso la migliore posizione visitata, mentre l'attrazione sociale verso la posizione migliore trovata da tutto lo sciame. Il moto è quindi rappresentabile mediante le seguenti equazioni:

$$V_i(t) = wV_i(t-1) + C_1R_1\left[X_{i_b}(t-1) - X_i(t-1)\right] + C_2R_2\left[X_{i_{gb}}(t-1) - X_i(t-1)\right] \quad (1)$$

$$X_i(t) = X_i(t-1) + V_i(t) \quad (2)$$

dove $i$ è riferito alla $i$-ma dimensione dello spazio di ricerca, $V$ è la velocità, $C_1$ e $C_2$ costanti positive, $w$ il coefficiente di inerzia, $X(t)$ la posizione, $X_b$ la posizione con migliore fitness visitata finora, $X_{gb}$ ("global best") il migliore punto trovato dall'intero sciame e $R_1$ e $R_2$ numeri casuali da una distribuzione uniforme in [0,1].
Fra le molte varianti [14], alcune modificano la topologia dello sciame [3,12]. La più tipica sostituisce $X_{gb}$ con $X_{lb}$ ("local best"), la migliore posizione trovata da un vicinato predefinito della particella. Ulteriori varianti si basano sulla topologia dei vicinati. In [12] sono valutate diverse topologie, fra cui ottengono buoni risultati topologie casuali e di Von Neumann, ma si evidenzia che il tipo di vicinato più efficiente dipende, in generale, dal problema. I due algoritmi detti "standard PSO" [2,10] utilizzano, rispettivamente, una topologia fissa ad anello e una "a stella stocastica", in cui ogni particella "informa" se stessa e $K-1$ vicini scelti in modo casuale. Tali relazioni sono inizializzate casualmente e aggiornate dopo ogni iterazione in cui il global best non è migliorato. Poiché il valore suggerito per $K$ è 3, questa versione è molto simile ad una con topologia fissa ad anello e due vicini per particella. Il riferimento per i nostri test è la versione descritta in [11].
Un altro aspetto critico per le prestazioni è la strategia di aggiornamento di $X_{gb}/X_{lb}$. La versione sincrona della PSO aggiorna le posizioni di tutte le particelle a turno in un'iterazione completa impropriamente detta "generazione": per ogni particella si calcolano velocità e posizione e si valuta la nuova fitness; il valore di $X_{gb}/X_{lb}$ si aggiorna solo al termine della generazione corrente.La versione asincrona invece aggiorna $X_{gb}/X_{lb}$ subito dopo la valutazione della fitness di ogni particella e lo sciame viene attratto più prontamente dai nuovi ottimi. Se la versione asincrona fosse anche distribuita, la struttura sequenziale si perderebbe e le equazioni di aggiornamento si potrebbero applicare a qualsiasi particella in qualsiasi momento, senza ordine specifico. Riguardo agli effetti di tale cambiamento, in [5] un algoritmo genetico evolve ordine e frequenza di aggiornamento delle particelle ottenendo spesso prestazioni migliori della PSO standard. Gli autori tuttavia notano anche che sono molte le caratteristiche (qualità

delle particelle, frequenza di aggiornamento, dimensione dello sciame ecc.) che influenzano le prestazioni.
Concentrandoci sulla sincronizzazione e sulla capacità di convergenza, misureremo la frequenza con cui si trovano gli ottimi assoluti in un benchmark come, ad es., quello descritto in [6].
Infatti, in [3] si mostra che, in generale, la versione sincrona richiede per convergere un numero di valutazioni di fitness maggiore di quella asincrona. Non è stato però discusso l'effetto della sincronizzazione sulla capacità di convergere ad una buona soluzione. Per valutarla abbiamo applicato le due politiche ad una stessa implementazione sequenziale.

## *2. Risultati*

I test hanno riguardato la versione Standard PSO 2006 (SPSO 2006) [10] sul 2009 Black Box Optimization Benchmark [6]; le funzioni sono descritte in [6,8] e la procedura sperimentale in [7]. Poiché ci interessano solo le variazioni nelle prestazioni dell'algoritmo, abbiamo usato solo funzioni separabili, sulle quali la PSO offre buone prestazioni [9]. Sono state valutate le topologie "global best" e "local best" utilizzando i parametri suggeriti in [10].
I risultati (Fig. 1) confermano le ipotesi sulla velocità di convergenza. I grafici mostrano il "tempo di esecuzione atteso" (ERT) delle quattro varianti sulle funzioni Sfera ed Ellissoide (unimodali) e Rastrigin (multimodale).
Se la dimensione dello spazio di ricerca è piccola, vi è solo una lieve differenza nel numero di valutazioni entro cui il valore obiettivo scende sotto la tolleranza richiesta.
Con "global best", tuttavia, all'aumentare della dimensione la strategia sincrona non è solo meno efficiente, ma anche meno efficace: sempre meno esecuzioni (numeri sopra il simbolo •) hanno successo e sempre più test (×) non raggiungono mai l'obiettivo; le prestazioni sono comunque paragonabili per le esecuzioni che terminano con successo (simboli +). Si nota bene il punto in cui gli ERT si allontanano dall'andamento lineare. Ciò è più evidente nei grafici della distribuzione dei tempi di esecuzione (Fig. 2, per maggiori dettagli vedi [7]) che riassumono le prestazioni dei quattro algoritmi su tutte le funzioni: a sinistra, con dimensione del problema pari a 5, le distribuzioni sono uguali, anche se la versione asincrona comunque risolve una funzione in più con la tolleranza più stretta (vedi legende).

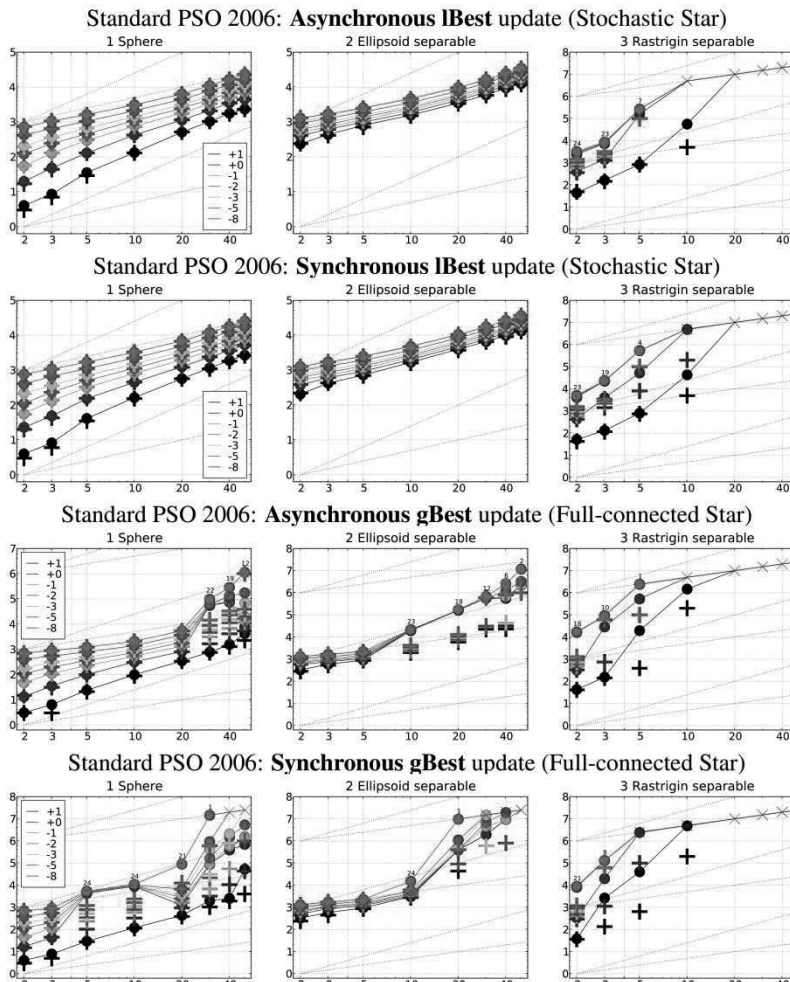



Figura 1: Tempo di esecuzione previsto (ERT, •) per raggiungere il valore obiettivo $f_{opt} + \Delta f$, $\Delta f = 10^k$ (k come da legende), e media ($log_{10}$) delle valutazioni della fitness nelle prove con successo (+) vs. dimensione del problema. L' $ERT\,(\Delta f)$ è pari a $\#FEs\,(\Delta f)$ diviso il numero di prove con successo. $\#FEs\,(\Delta f)$ è il numero totale di valutazioni della fitness per raggiungere $f_{opt} + \Delta f$ in tutti i tentativi (con e senza successo). Le croci (×) indicano i test senza successo. I numeri sopra i simboli • indicano il numero di prove con successo.

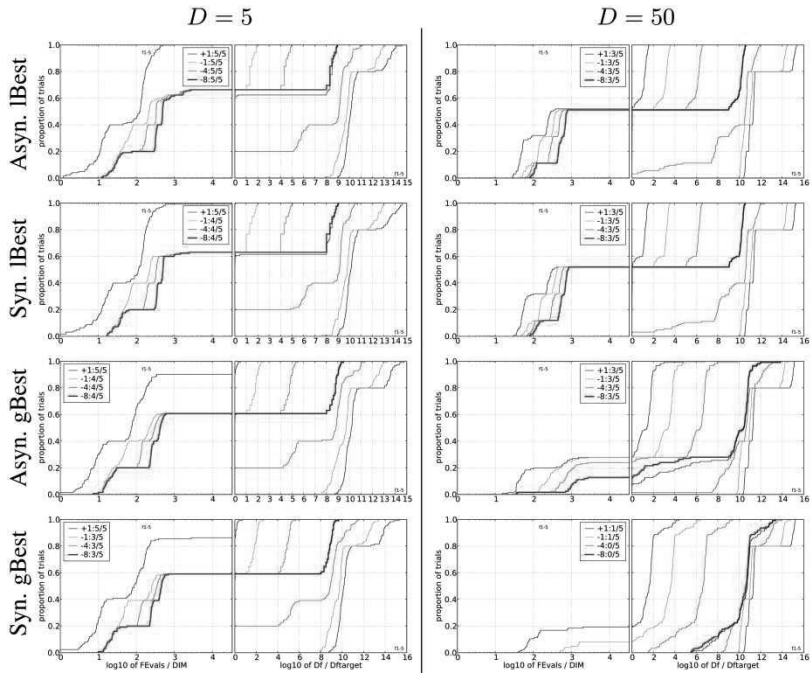



Figura 2: Distribuzione cumulativa empirica (ECDFs) del tasso di successo vs. tempo di esecuzione (sottografici a sinistra) o vs. $\Delta f$ (sottografici a destra). La linea più marcata indica i migliori risultati. Sottografici a sinistra: ECDF vs. numero di valutazioni della fitness per raggiungere $f_{opt} + \Delta f$ (con $\Delta f = 10^k$, $k$ primo valore in legenda), diviso per la dimensione $D$ dello spazio di ricerca. Sottografici a destra: ECDF dei migliori $\Delta f$ divisi per $10^k$ (in alto a sinistra in continuazione dei sottografici a sinistra), e migliori $\Delta f$ divisi per $10^{-8}$ per $D$, $10D$, $100D$ valutazioni di fitness (da destra a sinistra, curve nero, ciano e magenta). Ogni riga è relativa a una diversa combinazione di topologia (local/global best) e di strategia di comunicazione (sincrona/asincrona). Il secondo valore nelle legende è il numero di funzioni per le quali almeno una esecuzione ha avuto successo. $FEvals$ è il numero di valutazioni, $D$ o $DIM$ la dimensione dello spazio di ricerca e $Df$ o $\Delta f$ la distanza dall'ottimo globale.

Tabella 1: Frequenza di successo globale sulle funzioni separabili descritte in [8] vs. numero di particelle; per ogni combinazione problema/dimensione sono state eseguite 25 prove indipendenti, con valori di $DIM$ in {2,3,5,10,20,30,40,50}; una prova ha successo se la distanza tra global best e valore effettivo dell'ottimo scende sotto $10^{-8}$, altrimenti termina senza successo dopo $2 \cdot 10^4 \cdot DIM$ valutazioni della fitness.

| Numero Particelle | SPSO2006 | Sync-SPSO2006 | Async-gBest | Sync-gBest |
|---|---|---|---|---|
| $10 + 2\sqrt{DIM}$ | 0.647 | 0.648 | 0.485 | 0.373 |
| 32 | 0.704 | 0.713 | 0.587 | 0.447 |
| 64 | 0.727 | 0.729 | 0.673 | 0.520 |
| 100 | 0.729 | 0.731 | 0.686 | 0.547 |
| 128 | 0.730 | 0.732 | 0.691 | 0.568 |

Con un problema di dimensione 50, però, i grafici a destra mostrano come la PSO sincrona con topologia totalmente connessa sia molto meno efficace.
I test sono stati eseguiti con sciami relativamente piccoli: lo standard [10] indica infatti un numero di particelle $N_p = 10 + 2\sqrt{D}$ per problemi in $D$ dimensioni. In un'implementazione parallela il numero di particelle influenza poco il tempo di esecuzione. Come altera l'efficacia l'aumento di tale parametro? Per quantificare l'effetto della sincronizzazione anche rispetto al numero di particelle, Tab. 1 riporta una stima della probabilità globale di successo [1].
La PSO si conferma meno sensibile alla dimensione della popolazione [3] di altre metaeuristiche, ma molto sensibile al tipo di comunicazione [13]. Fig. 3 riassume le precedenti osservazioni e mostra che, per le due topologie, la probabilità di successo evolve diversamente all'aumentare della dimensione del problema, avendo fissato il numero di particelle e, soprattutto, la politica di aggiornamento.

## *3. Osservazioni*

Lo spazio non consente di discutere approfonditamente quanto osservato. Tuttavia, se la (1) è vista come operatore di ricombinazione [13], nella versione global best sincrona tutte le nuove soluzioni, nella stessa iterazione, vengono generate a partire da un "genitore" comune (il global best) che cambia solo dopo che tutte le particelle si sono spostate e hanno valutato la fitness.

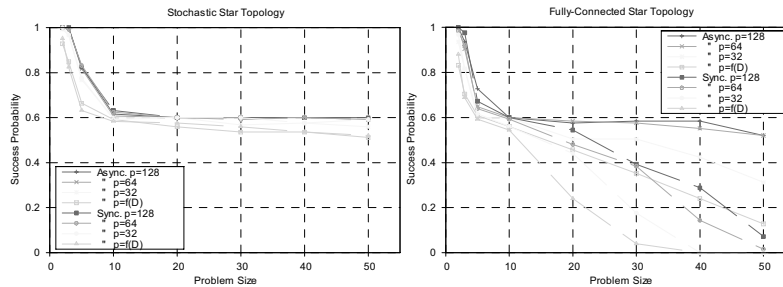



Figura 3: Frequenza di successo vs. dimensione D del problema. I tassi di successo sono valutati con i criteri adottati in Tab. 1. Le linee tratteggiate corrispondono all'aggiornamento sincrono, le linee continue all'aggiornamento asincrono; il colore è associato alla dimensione dello sciame.

Nella versione asincrona, invece, nella stessa iterazione particelle diverse sono influenzate da diversi attrattori sociali. Ciò può giustificare quanto visto poiché la perdita di diversità riduce la capacità di esplorazione di metaeuristiche basate su popolazione [15].
Poiché la PSO è molto sensibile alla topologia, si può dedurre che la diversità sia maggiormente preservata se le particelle interagiscono entro piccoli vicinati, indipendentemente dalla politica di aggiornamento. La SPSO2006, usata nei test come variante local best, utilizza una topologia casuale [4] in cui ogni particella deriva il proprio attrattore locale da un numero costante (default 3) di informatori, che la comprendono, creando una rete di comunicazione poco connessa che offre buone prestazioni (migliori della variante global best) anche nel caso sincrono.

## *4. Conclusioni*

I risultati suggeriscono che posporre l'aggiornamento degli attrattori alla fine di ogni generazione non pregiudichi l'efficacia della PSO se (e, apparentemente, solo se) si adotta una topologia poco connessa. In particolare, sincronizzare gli aggiornamenti degli attrattori con topologia totalmente connessa degrada le prestazioni all'aumentare

della dimensione del problema poiché riduce la diversità della popolazione e la capacità di ricerca dell'algoritmo.

### *Bibliografia*


1. A. Auger, N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. Proc. IEEE CEC 2005, Vol. 2, pp. 1777–1784, 2005.
2. D. Bratton, J. Kennedy. Defining a standard for particle swarm optimization. Proc. IEEE Swarm Intelligence Symposium, pp. 120–127, 2007.
3. A. Carlisle, G. Dozier. An off-the-shelf PSO. Proc. 2001 Workshop on PSO, pp. 1–6, 2001.
4. M. Clerc. Back to random topology. Tech. Report, 2007.
   URL: http://clerc.maurice.free.fr/pso/
5. L. Dioşan, M. Oltean. What else is evolution of PSO telling us? Journal of Artificial Evolution and Applications, pp. 1–12, 2008.
6. S. Finck, N. Hansen, R. Ros, A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Tech. report 2009/20, Research Center PPE, 2009.
7. N. Hansen, A. Auger, S. Finck, R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Tech. Report RR-6828, INRIA, 2009.
8. N. Hansen, S. Finck, R. Ros, A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Report RR-6829, INRIA, 2009.
9. N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger. PSO facing non-separable and ill-conditioned problems. Research Report RR-6447, INRIA, 2008.
10. J. Kennedy, M. Clerc. Standard PSO 2006.
    URL: http://www.particleswarm.info/Standard_PSO_2006.c
11. J. Kennedy and R. Eberhart. Particle swarm optimization. Proc. IEEE ICNN 1995, vol. IV, pp 1942–1948, 1995.
12. J. Kennedy, R. Mendes. Population structure and particle swarm performance. Proc. CEC 2002, pp. 1671–1676, 2002.
13. V. Miranda. Evolutionary algorithms with particle swarm movements. Proc. 13th Int. Conf. on Intelligent Systems Application to Power Systems, pages 6–21, 2005.
14. R. Poli, J. Kennedy, T. Blackwell. Particle swarm optimization: an overview. Swarm Intelligence, 1(1):33–57, 2007.
15. D. N. Wilke, S. Kok, A. A. Groenwold. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. Int. Journal for Numerical Methods in Engineering, 70(8):962–984, 2007.

# *La costruzione di memorie gerarchiche su grafi fattoriali*

**Francesco Palmieri**
*Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, Real Casa dell'Annunziata*, Via Roma 29, 81031 Aversa (CE)
francesco.palmieri@unina2.it

**Gianmarco Romano**
*Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, Real Casa dell'Annunziata*, Via Roma 29, 81031 Aversa (CE)
gianmarco.romano@unina2.it

**Pierluigi Salvo Rossi**
*Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, Real Casa dell'Annunziata*, Via Roma 29, 81031 Aversa (CE)
pierluigi.salvorossi@unina2.it

**Davide Mattera**
*Dipartimento di Ingegneria Biomedica, Elettronica e delle Telecomunicazioni, Università di Napoli "Federico II"*; Via Claudio, 21 80125 Napoli, Italia
mattera@unina.it

*Abstract:* In queste note viene presentato un algoritmo per la costruzione di un albero bayesiano fattoriale da un numero finito di esempi. Il grafo viene usato come struttura su cui operare inferenza bayesiana mediante propagazione delle probabilità. L'algoritmo è presentato in dettaglio con riferimento ad un esempio basato su un numero limitato di stringhe di caratteri.

*Parole chiave:* Grafi fattoriali; Memorie associative; Reti Bayesiane.

## *1. Introduzione*

La risoluzione di problemi probabilistici su grafi mediante il meccanismo della propagazione delle densità (*Belief Propagation, BP*) (Pearl, 1988; Loeligher et al., 2007), è un promettente paradigma per la gestione e il recupero dell'informazione in molti campi quali: l'elaborazione numerica dei segnali, le comunicazioni, l'intelligenza artificiale e la robotica. L'approccio bayesiano alla elaborazione dell'informazione consente di operare inferenze probabilistiche su dati incompleti, o su dati parzialmente affetti da errori, mediante il

cosiddetto meccanismo del "ragionamento bayesiano'': il sistema risponde con delle probabilità a posteriori dopo che i dati provenienti dalle osservazioni (evidenza) sono stati "iniettati" nel grafo. L'uso di questo meccanismo, che è alla base della decodifica di efficienti codici a correzione d'errore, quali i Turbo Codici (Forney, 2001), è sicuramente uno dei criteri più promettenti per la progettazione di sistemi di intelligenza artificiale che, addestrati su esempi reali, o su regole formali, forniscano predizioni sulla base di un *modello generativo* delle osservazioni. Questo paradigma più volte presentato nella letteratura sotto vari nomi, quali Reti Bayesiane, Modelli Generativi, Grafi Fattoriali, Campi di Markov, Memorie Associative e altro, riflette alcune intuizioni oramai molto diffuse (Jordan and Sejnowsky, 2001) su come un sistema informativo "neurale'' debba essere concepito. Le osservazioni e le ipotesi sul funzionamento della corteccia celebrale (Hawkins, 2004) sembrano confermare come la memoria sia una proprietà distribuita della rete neurale e che gli accessi ad essa siano multipli, distribuiti e bidirezionali. Nella costruzione di un sistema artificiale, che conservi le caratteristiche principali di un sistema neurale, si può immaginare che la risposta del sistema a degli stimoli sia una condizione di equilibrio informazionale[1] ottenuta come la combinazione probabilistica di inferenze parziali, provenienti da varie modalità sensoriali e da varie parti specializzate del sistema. Le ipotesi architetturali che emergono da queste considerazioni si discostano notevolmente dalla struttura seriale della macchina di Turing e promettono una rivisitazione completa della struttura dei sistemi di elaborazione numerica .

La teoria matematica della probabilità è il migliore strumento operativo per descrivere e progettare un sistema che deve poter apprendere, rispondere a delle domande in maniera determinata, e allo stesso tempo formulare delle ipotesi come risultato di una procedura di inferenza statistica (Jaynes, 2003).

La teoria delle Reti Bayesiane (Pearl, 1988), molto popolare in statistica, e ultimamente molto utilizzata nello studio delle sequenze di DNA in genetica, è il paradigma di riferimento per studiare le relazioni strutturate tra variabili aleatorie descritte su grafi. Diverse varianti operative sono state proposte nella letteratura, a seconda che le variabili aleatorie siano associate a dei nodi o a dei rami. In queste note abbiamo preferito utilizzate la versione basata sui Grafi Fattoriali (Factor Graphs, FG) in cui le variabili sono associate ai rami del grafo e i nodi sono i fattori (Forney, 2001; Loeliger, 2004; Loeliger et al., 2007). Nuove opportunità emergono nel modellare i sistemi con grafi fatto-

[1] I collegamenti tra la teoria dell'informazione e la termodinamica (Cover e Thomas, 1991) aprono delle prospettive interessanti in cui il sistema informativo assomiglia più ad una macchina termodinamica che ad una macchina sequenziale.

riali soprattutto nella possibilità di realizzare operazioni computazionalmente complesse in architetture distribuite basate su VLSI ed FPGA.

In queste note riportiamo un algoritmo per la costruzione gerarchica di una memoria associativa a partire da un numero finito di esempi. La memoria è un codice ad albero che si costruisce, mediante un'operazione di frazionamento gerarchico. Essa consente il richiamo associativo e la correzione degli errori mediante la propagazione delle probabilità sul grafo.

L'algoritmo è presentato con riferimento ad un esempio molto semplice: la memorizzazione e il richiamo di un piccolo insieme di parole. L'idea appare molto promettente ed è facilmente applicabile a database di grandi dimensioni per immagini, segnali monodimensionali e altre modalità sensoriali.

## *2. Costruzione del modello generativo*

Supponiamo che siano disponibili $n$ esempi $(X_1[j], X_2[j], \dots, X_N[j]), j = 1, .., n$ della stringa di variabili $(X_1, X_2, \dots, X_N)$, dove le variabili appartengono agli alfabeti $(A_1, A_2, \dots, A_N)$. La rappresentazione non codificata degli esempi richiede $n \sum_{i=1}^{N} log_2|A_i|$ bit di memoria, se ogni esempio è presente una sola volta, dove $|A_i|$ è la cardinalità dell'alfabeto $A_i$. L'accesso all'informazione richiede un indice di $log_2 n$ bit per identificare ogni esempio. Nella semplice memorizzazione degli esempi in forma di lista non si è sfruttata minimamente la struttura dell'informazione contenuta nelle stringhe. Le possibili relazioni locali tra le variabili non sono utilizzate. Le dipendenze locali possono essere molto utili al "richiamo associativo" quando solo informazione parziale, o errata, sulle stringhe è disponibile. L'obiettivo è pertanto cercare una rappresentazione distribuita che tenga conto delle caratteristiche strutturali delle stringhe e che possa essere usata come grafo fattoriale per inferenza probabilistica. L'idea di usare strutture efficienti di accesso a basi di dati è già contenuta nei cosiddetti *Hash Codes* (MacKay, 2003), molto usati in varie modalità nei sistemi operativi e nelle reti peer-to-peer. L'idea di spezzettare l'informazione viene qui usata in maniera diversa per trovare una rappresentazione gerarchica ad albero che costituisca il grafo fattoriale su cui operare dinamicamente mediante propagazione dell'inferenza.

Presentiamo l'idea con riferimento ad un semplice esempio. Si consideri l'insieme di 8 esempi della stringa $(X_1, X_2, \dots, X_8)$elencati in tabella

| $j$ | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ |
|---|---|---|---|---|---|---|---|---|
| 1 | L | A | S | C | I | A | T | E |
| 2 | O | G | N | I | b | b | b | b |
| 3 | S | P | E | R | A | N | Z | A |
| 4 | V | O | I | b | b | b | b | b |
| 5 | C | H | E | b | b | b | b | b |
| 6 | E | N | T | R | A | T | E | b |
| 7 | P | E | R | b | b | b | b | b |
| 8 | M | E | b | b | b | b | b | b |

Sono presenti 8 stringhe di caratteri, con il simbolo "b" che indica il "blank". Una prima analisi degli esempi mostra che le variabili hanno alfabeti diversi. In particolare

$A_1 = \{L, O, S, V, C, E, P, M\}$; $f_1 = \{1,1,1,1,1,1,1,1\}$;
$A_2 = \{A, G, P, O, H, N, E\}$; $f_2 = \{1,1,1,1,1,1,2\}$;
$A_3 = \{S, N, E, I, T, R, b\}$; $f_3 = \{1,1,2,1,1,1,1\}$;
$A_4 = \{C, I, R, b\}$; $f_4 = \{1,1,2,4\}$;
$A_5 = \{I, A, b\}$; $f_5 = \{1,2,5\}$;
$A_6 = \{A, N, T, b\}$; $f_6 = \{1,1,1,5\}$;
$A_7 = \{T, Z, E, b\}$; $f_7 = \{1,1,1,5\}$;
$A_8 = \{E, A, b\}$; $f_8 = \{1,1,6\}$;

dove abbiamo anche indicato il numero di occorrenze per ogni simbolo. La struttura degli esempi può essere sfruttata meglio se andiamo a codificare le sottostringhe. Per esempio, supponiamo di partizionare l'insieme in quattro coppie ottenendo le seguenti associazioni

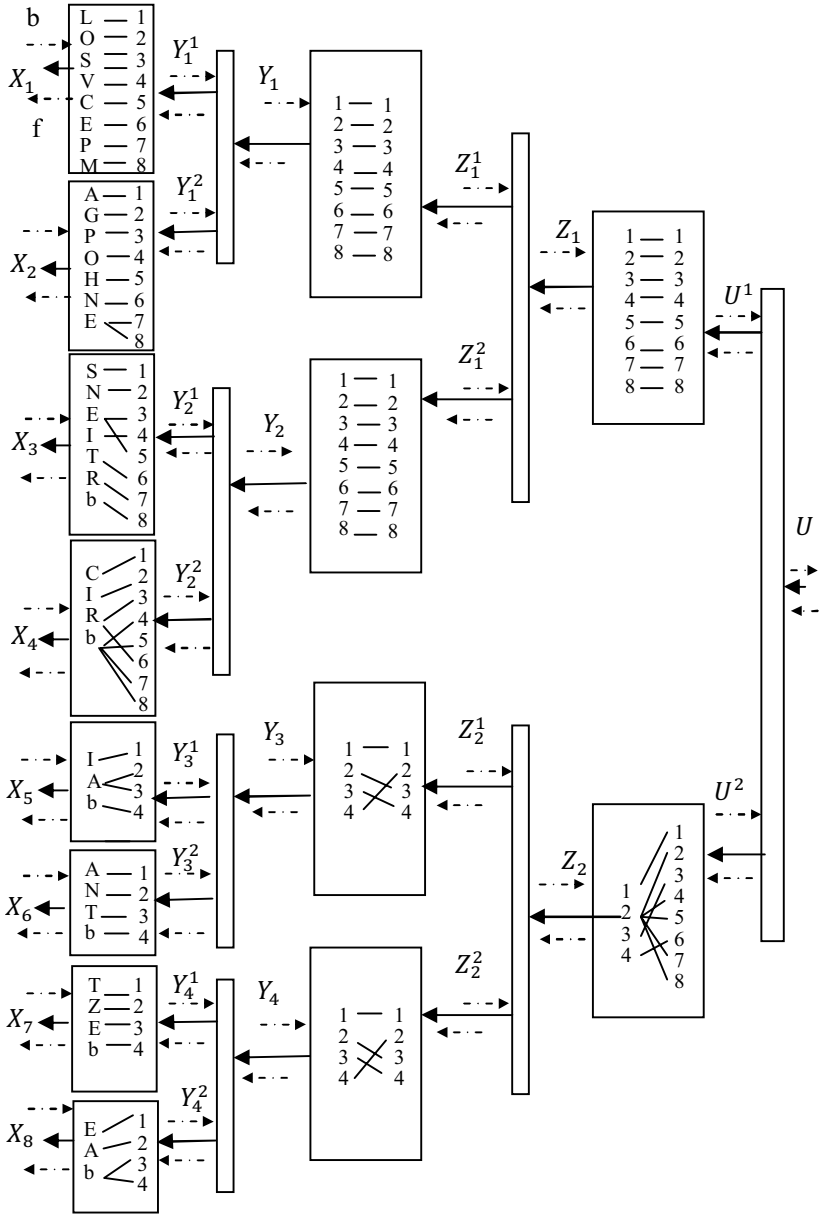
| $X_1X_2$ | $Y_1$ | $X_3X_4$ | $Y_2$ | $X_5X_6$ | $Y_3$ | $X_7X_8$ | $Y_4$ |
|---|---|---|---|---|---|---|---|
| $L\ A$ | 1 (1) | $S\ C$ | 1 (1) | $I\ A$ | 1 (1) | $T\ E$ | 1 (1) |
| $O\ G$ | 2 (1) | $N\ I$ | 2 (1) | $A\ N$ | 2 (1) | $Z\ A$ | 2 (1) |
| $S\ P$ | 3 (1) | $E\ R$ | 3 (1) | $A\ T$ | 3 (1) | $E\ b$ | 3 (1) |
| $V\ O$ | 4 (1) | $I\ b$ | 4 (1) | $b\ b$ | 4 (5) | $b\ b$ | 4 (5) |
| $C\ H$ | 5 (1) | $E\ b$ | 5 (1) | | | | |
| $E\ N$ | 6 (1) | $T\ R$ | 6 (1) | | | | |
| $P\ E$ | 7 (1) | $R\ b$ | 7 (1) | | | | |
| $M E$ | 8 (1) | $b\ b$ | 8 (1) | | | | |

dove abbiamo definito delle nuove variabili $(Y_1, Y_2, Y_3, Y_4)$ con alfabeti $B_1 = \{1,2,3,4,5,6,7,8\}$; $B_2 = \{1,2,3,4,5,6,7,8\}$; $B_3 = \{1,2,3,4\}$; $B_4 = \{1,2,3,4\}$. In parentesi sono anche indicate il numero di occorrenze. La fase gerarchica successiva è la determinazione degli alfabeti delle coppie delle nuove variabili

| $Y_1Y_2$ | $Z_1$ | $Y_3Y_4$ | $Z_2$ |
|---|---|---|---|
| 1 1 | 1 (1) | 1 1 | 1 (1) |
| 2 2 | 2 (1) | 4 4 | 2 (5) |
| 3 3 | 3 (1) | 2 2 | 3 (1) |
| 4 4 | 4 (1) | 3 3 | 4 (1) |
| 5 5 | 5 (1) | | |
| 6 6 | 6 (1) | | |
| 7 7 | 7 (1) | | |
| 8 8 | 8 (1) | | |

in cui due nuove variabili $(Z_1, Z_2)$ con alfabeti $B_1 = \{1,2,3,4,5,6,7,8\}$; $B_2 = \{1,2,3,4\}$ sono state definite. La tabella finale rappresenta l'alfabeto di $Z_1Z_2$ con la associazione alla variabile $U$ di alfabeto $C = \{1,2,3,4,5,6,7,8\}$

| $Z_1Z_2$ | $U$ |
|---|---|
| 1 1 | 1 (1) |
| 2 2 | 2 (1) |
| 3 3 | 3 (1) |
| 4 2 | 4 (1) |
| 5 2 | 5 (1) |
| 6 4 | 6 (1) |
| 7 2 | 7 (1) |
| 8 2 | 8 (1) |

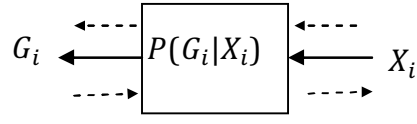

b
f
$X_1$
$X_2$
$X_3$
$X_4$
$X_5$
$X_6$
$X_7$
$X_8$
$Y_1^1$
$Y_1^2$
$Y_2^1$
$Y_2^2$
$Y_3^1$
$Y_3^2$
$Y_4^1$
$Y_4^2$
$Y_1$
$Y_2$
$Y_3$
$Y_4$
$Z_1^1$
$Z_1^2$
$Z_2^1$
$Z_2^2$
$Z_1$
$Z_2$
$U^1$
$U^2$
$U$

La figura precedente mostra il grafo generativo completo degli esempi in cui sono anche indicate le tabelle di corrispondenza. Ovviamente le partizioni delle variabili utilizzate ai vari livelli gerarchici sono solo una delle possibili associazioni. Si potrebbero considerare delle terne, o delle variabili che si congiungono a livelli gerarchici successivi. Nell'esempio di figura si vede chiaramente come ad un unico valore della variabile $U$, che costituisce una sorta di *primitiva*, viene associata (richiamata) un'unica stringa.
Per esempio, per $U = 5$, $(Z_1 Z_2) = (5,2)$, $(Y_1, Y_2, Y_3, Y_4) = (5,5,4,4)$, $(X_1, X_2, \dots, X_8) = (C, H, E, b, b, b, b, b)$.

*Gli esempi sono stati "cablati" nell'albero.*

Più interessante è l'uso del grafo come memoria associativa in cui l'albero viene usato come grafo fattoriale bayesiano. I blocchi rappresentano le funzioni di probabilità condizionata (in questo caso le corrispondenze da sinistra a destra sono funzioni deterministiche) tra le variabili aleatorie e i messaggi possono essere propagati in entrambe le direzioni sull'albero per inferenza probabilistica. Illustriamo anche questa operazione con un esempio. Supponiamo che alla fine dell'albero sia disponibile la stringa incompleta (E?????E?) e vogliamo usare la memoria per trovare la migliore associazione. Denotiamo con $bX$ e $fX$, i messaggi inverso (*backward*) e diretto (*forward*) rispettivamente, per la variabile generica $X$. Le regole per la propagazione delle densità sono semplici somme e prodotti (*sum-product*) descritte nella letteratura (Loeliger, 2004) (vedi anche (Palmieri, 2008) per una semplice formulazione matriciale). Poiché $X_1$ e $X_7$ sono note, $bX_1 = (0,0,0,0,0,1,0,0)$ e $bX_7 = (0,0,1,0)$. Le altre variabili sono sconosciute, pertanto $bX_2 = bX_3 = Unif(7)$, $bX_4 = Unif(4)$, $bX_5 = Unif(3)$, $bX_6 = Unif(4)$, $bX_8 = Unif(3)$, dove $Unif(n)$ indica la distribuzione uniforme su $n$ simboli. Proseguendo nell'albero fino alla radice e ripropagando in avanti i messaggi abbiamo la ricostruzione completa (E,N,T,R,A,T,E,b).
Il richiamo associativo può essere completato anche senza arrivare alla radice dell'albero poiché i messaggi rientranti dai rami laterali possono già risolvere alcune delle incertezze locali.

Molto utile è anche l'uso della memoria in presenza di errori, ovvero quando si presentano al sistema delle stringhe contenenti dei caratteri sbagliati. Per trattare tale eventualità bisogna modellare anche il processo di generazione dell'errore definendo delle nuove variabili $G_i, i = 1, \dots, 8$



dove $P(G_i|X_i)$ è la matrice $|A_i| \times |A_i|$ delle probabilità condizionate che modellano lo scambio dei caratteri. Supponendo qui per semplicità che l'errore possa avvenire in maniera uniforme su tutti i simboli dell'alfabeto per ogni variabile, la matrice $P(G_i|X_i)$ conterrà la probabilità di corretta generazione $p_c$ sulla diagonale e le probabilità di scambio $(1 - p_c)/(|A_i| - 1)$ uniformemente distribuite sulle righe. Il richiamo associativo quindi parte dalle $G_i$e presegue inizialmente a ritroso nell'albero, per poi essere ripropagato in avanti per produrre una migliore inferenza sulle $X_i$ mediante $fX_i$.
Per meglio illustrare il processo associativo, proponiamo il seguente esempio. Supponiamo che $p_c = .9$ e che la stringa osservata sia (EHTRATbb). Ci sono due errori che vorremmo correggere mediante richiamo associativo. Propagando a ritroso nel blocco che modella gli errori otteniamo
$bX_1 = (.0143, .0143, .0143, .0143, .0143, .9, .0143, .0143)$,
$bX_2 = (.0167, .0167, .0167, .0167, .9, .0167, .0167)$,
$bX_3 = (.0167, .0167, .0167, .0167, .9, .0167, .0167)$,
$bX_4 = (.0333, .0333, .9, .0333)$,
$bX_5 = (.05, .9, .05)$,
$bX_6 = (.0333, .0333, .9, .0333)$,
$bX_7 = (.0333, .0333, .0333, .9)$,
$bX_8 = (.05, .05, .9)$. Operando la propagazione a ritroso e poi di nuovo verso le foglie otteniamo l'inferenza finale su $X_1, X_2, \ldots, X_8$ mediante i prodotti normalizzati $pX_i = fX_i * bX_i, i = 1, \ldots, 8$, che danno (ENTRATEb).

## *3. Conclusioni*

L'algoritmo proposto in queste note mostra come sia possibile costruire un grafo gerarchico a partire dalle caratteristiche congiunte di variabili presenti in un numero finito di esempi. La rappresentazione distribuita consente di integrare conoscenze parziali e globali sui dati mediante flussi probabilistici bidirezionali.

Le opportunità offerte dal paradigma presentato sono eccellenti. Esso può essere facilmente applicato a scenari di utilizzo molto diversi quali la elaborazione delle immagini, l'analisi dei segnali e il controllo adattativo.

## *Bibliografia*


1. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley (1991).
2. Forney, G.D. Jr.: Codes on graphs: normal realizations. IEEE Trans. Information Theory. Vol. 47, no. 2, pp. 520-548 (2001).
3. Hawkins, J.: On Intelligence. Times Books (2004).
4. Jaynes E.T.: Probability Theory: The Logic of Science. Cambridge University Press (2003).
5. Jordan, M.I., Sejnowski, T.J. eds.: Graphical Models: Foundations of Neural Computation. MIT Press (2001).
6. Loeliger, H.A.: An Introduction to Factor Graphs. IEEE Signal Processing Magazine. pp. 28-41, Jan (2004).
7. Loeliger, H.A., Dauwels, J., Hu, J., Korl, S., Ping, L., Kschischang, F.R: The Factor Graph Approach to Model-Based Signal Processing. Proceedings of the IEEE. Vol. 95, N. 6, pp. 1295-1322, June (2007).
8. MacKay, D.J.C.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press (2003).
9. Palmieri, F.: Notes on Factor Graphs, New Directions in Neural Networks with IOS Press in the KBIES book series. Proceedings of WIRN 2008, Vietri sul mare. June (2008).
10. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. 2nd ed. San Francisco: Morgan Kaufmann, (1988).

# *Motivazione ed emozione in organismi artificiali*

**Domenico Parisi**
*Istituto di Scienze e Tecnologie della Cognizione - CNR - Roma*
Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: +39 06-44595333
domenico.parisi@istc.cnr.it

**Giancarlo Petrosino**
*Istituto di Scienze e Tecnologie della Cognizione - CNR - Roma*
Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: +39 06-44595333
giancarlo.petrosino@istc.cnr.it

## *1. Introduzione*

Il comportamento tende ad essere visto come risposte motorie agli stimoli sensoriali ma questo modo di vedere il comportamento è incompleto e sviante. Immaginiamo un organismo che vede davanti a sé del cibo. L'organismo risponderà a questo stimolo avvicinandosi e mangiando il cibo solo se l'organismo ha fame. Se non ha fame, semplicemente ignorerà il cibo e farà qualche altra cosa. Questo ci dice che gli stimoli ambientali non sono la sola causa del comportamento, e nemmeno la causa principale. Per spiegare il comportamento di un organismo dobbiamo prendere in considerazione lo stato motivazionale in cui l'organismo sta in quel momento. Lo stato motivazionale, insieme agli stimoli provenienti dall'ambiente, ci permettono di prevedere quello che farà l'animale. Gli stimoli da soli non bastano.

Il problema è che un organismo può avere molte motivazioni diverse (mangiare, bere, cercare un partner sessuale, scappare da un predatore, ecc.) ma in ogni particolare momento può agire per soddisfare solo una motivazione. C'è bisogno quindi di un livello strategico del comportamento in cui viene scelta la motivazione da perseguire in ogni particolare momento, e poi di un livello tattico in cui vengono eseguiti i comportamenti necessari a soddisfare la motivazione decisa al livello strategico. Ogni motivazione ha in ogni dato momento un certo livello

di intensità e la motivazione che viene scelta è quella che in quel momento ha il livello di intensità più alto.

## *2. Esperimenti*

Una popolazione di organismi artificiali vive in un ambiente in cui ci sono, sparsi a caso, elementi di cibo e elementi d'acqua (Saglimbeni & Parisi, 2009). Per sopravvivere e riprodursi un organismo ha bisogno sia di mangiare che di bere. L'organismo ha due serbatoi nel proprio corpo, uno di energia e l'altro di acqua, una data quantità di energia e di acqua viene consumata ad ogni istante di tempo, e se uno dei due serbatoi rimane vuoto, l'organismo muore. Il comportamento degli organismi è controllato da una rete neurale artificiale con unità sensoriali che codificano la posizione degli elementi di cibo e di acqua più vicini, unità motorie che codificano i movimenti dell'organismo nell'ambiente, e uno strato intermedio di unità interne. I pesi delle connessioni della rete neurale sono evoluti usando un algoritmo genetico il quale seleziona per la riproduzione gli organismi che vivono più a lungo e che trasmettono ai loro figli i loro stessi pesi con qualche variazione casuale (mutazioni genetiche).

Ci sono tre popolazioni di organismi che vivono in tre ambienti diversi. Nell'ambiente 1 (A1) cibo e acqua sono ugualmente abbondanti, nell'ambiente 2 (A2) c'è più cibo che acqua (o viceversa), mentre l'ambiente 3 (A3) è stagionale, cioè è un ambiente in cui si alternano stagioni con più cibo che acqua e stagioni con più acqua che cibo. Le simulazioni cominciano assegnando a caso i pesi della rete neurale degli organismi, che quindi non si comportano in modo molto efficiente, ma nel corso delle generazioni la riproduzione selettiva dei migliori individui e l'aggiunta delle variazioni casuali ai pesi ereditati si traducono in comportamenti più efficienti che fanno sì che gli organismi vivano più a lungo. Per tutte e tre le popolazioni facciamo due versioni della simulazione, una in cui la rete neurale è quella che abbiamo descritto e l'altra in cui la rete neurale ha altre unità di input sensoriale, questa volta che informano la rete neurale non sull'ambiente esterno, ma sullo stato del corpo dell'animale, cioè sul livello corrente di energia e di acqua nei due serbatoi.

I risultati mostrano che in A1 e in A2 il fatto che il "cervello" degli organismi sia informato sullo stato corrente del corpo non dà nessun particolare vantaggio. Il comportamento che evolve in A1 (stessa abbondanza di cibo e acqua) è quello di avvicinarsi e ingerire l'elemento

più vicino, che può essere cibo o acqua, e questo assicura che il livello di energia e di acqua nel corpo dell'organismo non vada a zero. E' quello che l'organismo vede in quel momento che determina quale motivazione (mangiare o bere) è quella dominante e determina il comportamento dell'organismo. In A2 (cibo costantemente più abbondante dell'acqua, o viceversa) gli organismi evolvono il comportamento di avvicinarsi all'elemento meno abbondante nell'ambiente, ignorando quello più abbondante a meno che non sia molto vicino. L'ambiente in cui la popolazione si è evoluta ha determinato una maggiore importanza di una motivazione rispetto all'altra e la decisione a livello strategico su che cosa fare viene presa in base a questa importanza. In questi due ambienti il "cervello" degli organismi non ha bisogno di essere informato sullo stato corrente del corpo per prendere le decisioni giuste, cioè l'organismo non ha bisogno di sentire fame o sete. Invece in A3 (l'ambiente stagionale) gli organismi che sono informati su quanta energia e quanta acqua c'è nel corpo, cioè che sentono fame e sete, vanno meglio di quelli che non sono informati. Questo si spiega con il fatto che l'informazione proveniente dall'ambiente esterno non è sufficiente a decidere se farsi guidare dalla motivazione di mangiare o da quella di bere.

Quelle che abbiamo descritte fino ad ora sono simulazioni del livello strategico del comportamento, quello della scelta motivazionale. Il fatto che un organismo sia guidato nel suo comportamento dalla motivazione decisa al livello strategico non implica necessariamente che l'organismo provi emozioni o si trovi in particolari stati emotivi. Le emozioni sono collegate con le motivazioni ma sono una cosa distinta dalle motivazioni (anche se vengono spesso confuse con le motivazioni). Che cosa sono allora gli stati emotivi? La nostra ipotesi è che gli stati emotivi sono stati in cui si trova un organismo e che sono emersi evolutivamente per rendere il meccanismo strategico della decisione motivazionale più efficiente, meno soggetto ad errori e più veloce. Gli organismi che hanno molte motivazioni differenti, che debbono prendere in considerazione molti fattori interni o esterni nel decidere quale motivazione debba guidare il loro comportamento in ogni dato momento, che debbono prendere decisioni rapide, o che debbono decidere se persistere o abbandonare una motivazione che l'organismo non riesce a soddisfare, possono avere meccanismi di decisione motivazione poco efficienti, soggetti ad errore, lenti, e nel complesso incapaci di assicurare la sopravvivenza e la possibilità di riprodursi dell'organismo. E' a questo punto che entrano in gioco gli stati emotivi, che sono stati dell'organismo che modificano l'intensità delle diverse motivazioni in modo che il meccanismo di decisione motivazionale funzioni meglio.

Facciamo una nuova versione delle tre simulazioni già descritte negli ambienti A1, A2 e A3. Tutte e tre le popolazioni hanno un "cervello" che viene informato dello stato corrente del corpo (livello di energia e di acqua nei due serbatoi) ma in più ora hanno un nuovo circuito nel loro cervello (circuito emotivo) che collega le unità che codificano il livello di energia e di acqua nel corpo dell'organismo con le unità motorie passando attraverso un secondo strato di unità interne. Quali sono i risultati di queste simulazioni? Di nuovo, in A1 e in A2 non fa differenza se gli organismi hanno questo circuito emotivo oppure no. Invece in A3, l'ambiente stagionale, gli organismi che hanno il circuito emotivo vanno meglio di quelli che non ce l'hanno. L'esistenza di un circuito emotivo rende il meccanismo di decisione motivazionale più efficiente e questo si traduce in una vita che dura più a lungo. Ad esempio, quando tra il livello di energia nel corpo è più basso di quello dell'acqua e tutti e due i livelli sono abbastanza vicini allo zero, per cui la vita dell'organismo è in pericolo, il circuito emotivo fa "alzare la voce" alla fame rispetto alla sete, e questo garantisce che l'animale non faccia errori e cerchi il cibo piuttosto che l'acqua, salvando così la propria vita. Una controprova della funzione del circuito emotivo si ha lesionando tale circuito. Gli organismi con il circuito lesionato si comportano meno bene non solo degli organismi con circuito emotivo integro ma anche degli organismi che non hanno mai avuto un circuito emotivo.

## *Bibliografia*

1. Canamero, L., Emotion understanding from the perspective of autono - mous research, in *Neural Networks*, Vol. 18, pp. 445-455, 2005.
2. Cecconi, F. & Parisi, D., Neural networks with motivational units, in *From animals to animats 2: Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior*, Cambridge, Mass., MIT Press, pp. 346-355, 1993.
3. Chajut, E. & Algon, C., Selective attention under stress: implications for theories of social cognition, in *Journal of Personality and Social Psychology*, 85, 231-248, 2003.
4. Davis, D.N., Agents, emergence, emotion and representation, in *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON2000)*, 2000.

5. Davis, D.N., Linking perception and action through motivation and affect, in *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 20, pp. 37-60, 2008.
6. Parisi, D., Motivation in artificial organisms, in *Machine learning and perception*, Singapore, World Scientific, pp. 3-19, 1996.
7. Petrosino, G. & Parisi, D., Deciding whether to look for food or to avoid predators. In preparation
8. Ruini, F. & Parisi, D. (2008). Selective attention in artificial organisms (abstract). In *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems* (p. 799), Cambridge, MA, MIT Press.
9. Saglimbeni, F. & Parisi, D., Input from the external environment and input from within the body, in *Proceedings of the 2009 IEEE European Conference on Evolutionary Computation,* in press.

# *Clustering di attrattori di reti booleane casuali*

**Andrea Roli**
*DEIS-Cesena, Alma Mater Studiorum Università di Bologna*;
Via Venezia 52, 47521 Cesena, Italia;
andrea.roli@unibo.it

**Stefano Benedettini**
*DEIS-Cesena, Alma Mater Studiorum Università di Bologna*;
Via Venezia 52, 47521 Cesena, Italia;
s.benedettini@unibo.it

**Roberto Serra**
*Dipartimento di scienze sociali, cognitive e quantitative,*
*Università di Modena e Reggio Emilia;*
Via Allegri 9, 42121 Reggio Emilia, Italia
roberto.serra@unimore.it

## *1. Introduzione*

Le reti booleane (RB) sono state introdotte da Kauffman (1993) come modello per lo studio della logica e della dinamica delle reti di regolazione genica e come astrazione di sistema complesso, per studiare le caratteristiche dell'evoluzione dei sistemi viventi. In particolare, una delle principali congetture riguardo la relazione tra RB e reti di regolazione genica consiste nell'interpretare gli attrattori della rete come tipi cellulari.

In questo contributo si presentano i risultati di un'analisi sperimentale volta a studiare la distribuzione delle distanze tra gli attrattori di una RB al variare dei regimi dinamici ordinato, critico e caotico. L'obiettivo a lungo termine di questa ricerca è duplice: da una parte si vogliono studiare approfonditamente le caratteristiche delle RB come modello di dinamica cellulare; dall'altra, la conoscenza di come la traiettoria di una RB possa moversi da un attrattore all'altro, apre la possibilità di progettare RB per il controllo di sistemi autonomi.

## *2. Reti booleane casuali*

Una RB è costituita da un grafo diretto di N nodi, a ciascuno dei quali sono associate una variabile booleana e una funzione (booleane) di transizione. Gli argomenti della funzione di transizione di un nodo sono le variabili associate ai nodi origine degli archi incidenti. Lo stato di una RB è definito dalla N-pla di valori delle variabili $(x_1,...,x_N)$. La dinamica di aggiornamento della rete può essere di vario tipo. In questo articolo si considererà una dinamica tempo-discreta e una forma di aggiornamento sincrono, in cui l'aggiornamento dei nodi avviene in parallelo ad ogni istante di tempo. In questo modo la dinamica della rete è deterministica e la traiettoria compiuta dal sistema nello spazio degli stati può essere decomposta in un transitorio e un attrattore, che può consistere in un punto fisso o un ciclo di periodo maggiore di 1.

Notevole interesse rivestono RB con topologia e funzioni casuali (RBC). In particolare, in questo contributo si considerano reti di N nodi con K ingressi (senza autoanelli) distribuiti uniformemente e funzioni booleane definite assegnando ad ogni possibile valore della tabella di verità un 1 con probabilità p e uno 0 con probabilità 1-p. Dato un valore di p (detto bias o parametro di omogeneità), le RBC mostrano una transizione tra regime ordinato e caotico, separato da un valore critico di connettività $K_c$ (Derrida e Pomeau, 1986). La relazione tra K e p che mantiene la rete nella fase critica è data da: $K = 2p\ (1 \square p)^{-1}$. Le RBC critiche rivestono particolare interesse poiché esibiscono il miglior compromesso tra robustezza e adattatività (Aldana et al., 2006).

## *3. Attrattori nelle RBC*

L'interpretazione degli attrattori delle RBC come tipi cellulari, suggerisce di studiare le relazioni di somiglianza e le proprietà di raggiungibilità tra gli attrattori di una rete. La conoscenza di queste caratteristiche permette di formulare ipotesi sperimentali più precise per verificare le congetture formulate e apre prospettive per l'impiego di RBC come modelli della dinamica cellulare (per esempio, cellule staminali e tumorali) e come dispositivi di controllo in sistemi autonomi. In questo contributo si presentano i risultati di un'analisi sperimentale volta a stimare la distribuzione delle distanze tra gli attrattori di una rete.

Per lo studio degli attrattori è importante distinguere tra *spazio delle configurazioni* e *spazio degli stati*. Il primo è costituito dall'insieme di

tutti i possibili stati della rete, quindi un insieme di cardinalità $2^N$, sul quale possono essere definite distanze di vario genere (per esempio, la distanza di Hamming). Lo spazio degli stati invece, è definito dallo spazio delle configurazioni al quale si aggiunge la topologia indotta dalle transizioni tra gli stati della rete, dipendenti dalle funzioni booleane.

In questo contributo si considereranno gli attrattori come sottoinsiemi dello spazio delle configurazioni e la distanza tra due attrattori $A_i$ e $A_j$ è definita come segue:

$$d(A_i , A_j) = \min \{ H_d (s, s') : s \square A_i , s' \square A_j \}$$

dove $H_d (s, s')$ è la distanza di Hamming tra le configurazioni s e s'.

La funzione testè definita può essere considerata una stima della distanza tra due attrattori poiché indica in numero minimo di nodi da cambiare in un attrattore affinché la traiettoria del sistema prosegua direttamente sul secondo attrattore. È naturalmente possibile definire anche altre distanze, basate sempre sulla distanza di Hamming o su altre distanze, quali, ad esempio, la Normalized Compression Distance.

In questo sudio siamo interessati ad analizzare le proprietà della distribuzione degli attrattori di una rete in funzione della distanza in modo da avere un quadro della distribuzione "spaziale" degli attrattori.

## *4. Metodologia*

L'analisi che verrà presentata riguarda il campionamento di RBC rispettivamente di 50, 60 e 70 nodi, con K = 3 e valori di p relativi alla fase ordinata e caotica e lungo la linea critica. In particolare, sono generate reti con i seguenti valori di p: 0.211324, 0.788675 (valori critici), 0.85 (rete in regime ordinato), 0.5 (rete in regime caotico). Per ogni configurazione di parametri, sono state generate 50 realizzazioni di rete. La dinamica di ogni rete è stata simulata a partire da $10^5$ stati iniziali casuali in modo da campionarne gli attrattori. Dati gli attrattori della rete, può essere costruita una matrice delle distanze in funzione della distanza definita in precedenza. La matrice delle distanze è stata poi utilizzata per due tipi di analisi: (i) la distribuzione del coefficiente di clustering (pesato) e (ii) la generazione di dendrogrammi.

*4.1 Coefficiente di clustering*

Il coefficiente di clustering $C_i$ del nodo *i* di un grafo fornisce una misura della tendenza dei vicini del nodo a formare un grafo completamente connesso. Dato un grafo con archi non pesati, il coefficiente di clustering $C_i$ è pari al valore massimo 1 se i vicini del nodo *i* formano un sottografo completamente connesso, mentre vale 0 se i vicini di *i* non hanno alcun collegamento. La media dei coefficienti di clustering dei nodi di un grafo fornisce dunque una stima di quanto il grafo sia caratterizzato da raggruppamenti (cluster) di nodi. Formalmente, il coefficiente di clustering di una rete è pari a:

$$C = \frac{1}{N}\sum_{i=1}^{N} C_i \qquad\qquad C_i = n_i/g_i$$

essendo $n_i$ il numero di connessioni tra i vicini del nodo i, e $g_i$, il numero massimo possibile di tali connessioni. È possibile estendere la definizione di coefficiente di clustering anche a grafi con archi pesati (Zangh e Horvath, 2005); in questo caso, maggiore è il peso di un arco, maggiore è l'intensità di collegamento tra i due nodi. I valori utilizzati per il calcolo di questo parametro sono riferiti alla matrice di adiacenza della rete, in cui l'elemento $a_{ij}$ corrisponde al peso dell'arco che ha origine in *i* e destinazione in *j*; $a_{ij} = 0$ se i = j oppure se l'arco (i,j) non è presente. In formule:

$$n_i = \frac{1}{2}\sum_{u\neq i}\sum_{\{v|v\neq i, v\neq u\}} a_{iu}a_{uv}a_{vi}$$

$$g_i = \frac{1}{2}((\sum_{u\neq i} a_{iu})^2 - \sum_{u\neq i} a_{iu}^2)$$

Nella nostra analisi il peso di un arco che collega due nodi (attrattori) è l'inverso della distanza tra gli attrattori.

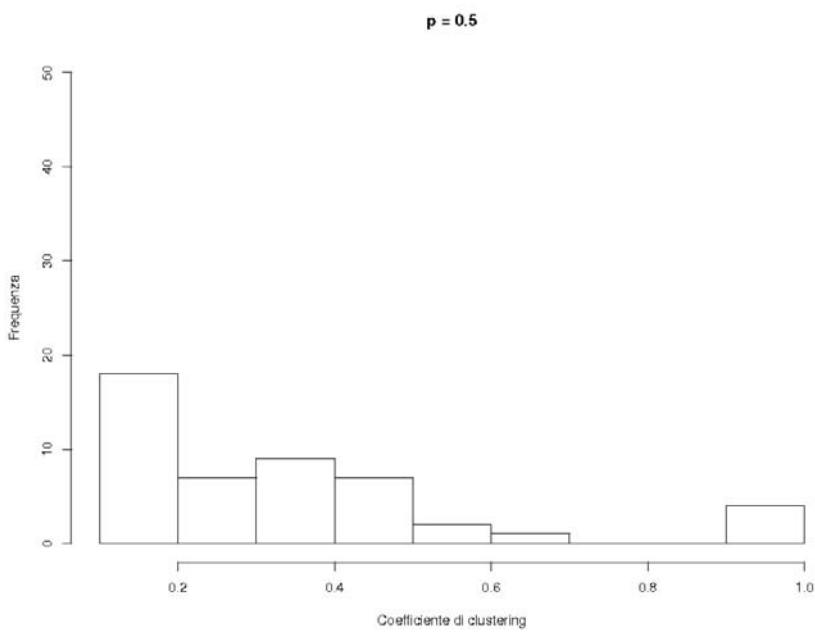
*4.2 Dendrogrammi*

La matrice delle distanze relativa agli attrattori di una rete può essere utilizzata anche per rappresentare graficamente la distribuzione in cluster degli attrattori. Per ogni rete è stato generato un dendrogram-

ma, che rappresenta in una sola struttura di dati i possibili raggruppamenti degli elementi di un insieme. L'analisi dei dendrogrammi degli attrattori permette di avere una rappresentazione visiva della tendenza degli attrattori a formare cluster. I risultati che saranno presentati si riferiscono a dendrogrammi costruiti con l'algoritmo "single-link" (Jain et al, 1999).

## *5. Analisi sperimentale*

In questa sezione si presentano i risultati sperimentali ottenuti simulando le reti tramite "The Boolean Network Toolkit" (Benedettini, 2009). Per motivi di limiti di spazio, si presenteranno solo gli istogrammi relativi alla distribuzione del coefficiente di clustering per N = 70 e p = 0.5, 0.788675, 0.85 (figure 1, 2 e 3 rispettivamente), che rappresentano un esempio titpico dei risultati ottenuti.



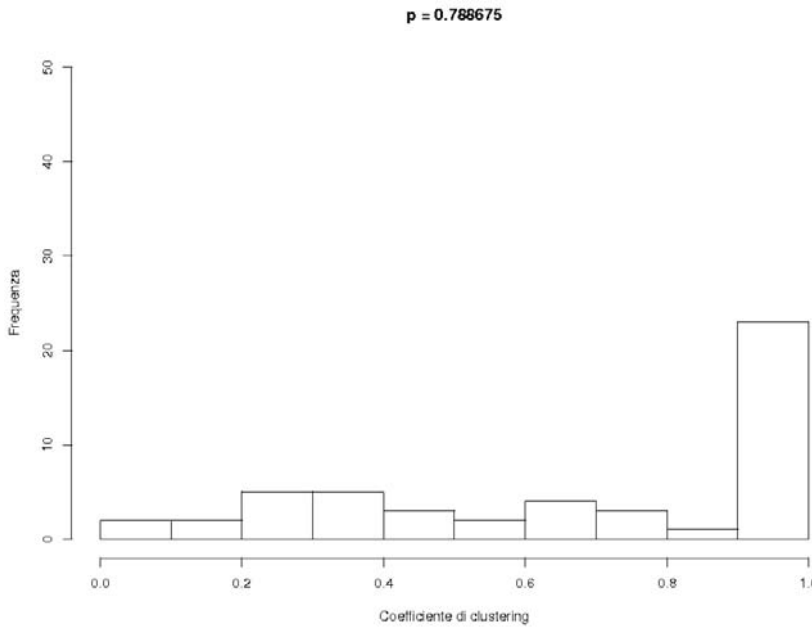



Figure 1 e 2: Distribuzione dei valori di coefficiente di clustering pesato relativo alla matrice delle distanze tra gli attrattori in un campione di 100 reti con N = 70, K = 3 e p = 0.5 e 0.788675, rispettivamente.

Come si può osservare dalle figure, la distribuzione del coefficiente spazia per tutto l'intervallo [0,1], indipendentemente dalla configurazione dei parametri caratteristici della rete. Questo significa che nell'insieme di reti generate a partire da una data configurazione dei parametri K e p, è sempre possibile trovare reti con un dato valore per C. Altro aspetto di rilievo è che la distribuzione di C è marcatamente centrata verso lo 0 per reti caotiche, mentre ha un picco verso il massimo per reti critiche e ordinate. Da notare che per le reti critiche C segue una distribuzione con una coda più spessa verso lo 0 rispetto a quella delle reti ordinate.

La considerazione principale che segue queste analisi è che reti critiche e ordinate hanno una maggiore probabilità di presentare un insieme degli attrattori in uno o più gruppi, mentre reti caotiche presentano con maggiore frequenza una distribuzione sparsa degli attrattori.

Questa osservazione porta a domandarsi quali siano le distanze misurate tra gli attrattori. Le figure 4, 5 e 6 riportano i dendrogrammi di clustering tipici per reti delle stesse famiglie di quelle considerate in precedenza: l'asse delle ordinate indica la distanza alla quale è stata eseguita ogni operazione di fusione tra due cluster. In particolare, i valori estremi indicano le distanze minima e massima tra gli attrattori della rete. Dagli esperimenti effettuati è risultato che le reti caotiche presentano una distanza mediana più elevata di quella delle reti critiche e ordinate. Altra osservazione importante, che conferma i risultati discussi precedentemente in merito al coefficiente di clustering, è che le reti caotiche presentano attrattori la cui distribuzione "spaziale" (funzione della distanza definita) appare sparsa, senza specifiche strutture a cluster, come invece è tipico in reti critiche e, all'estremo, in reti ordinate.

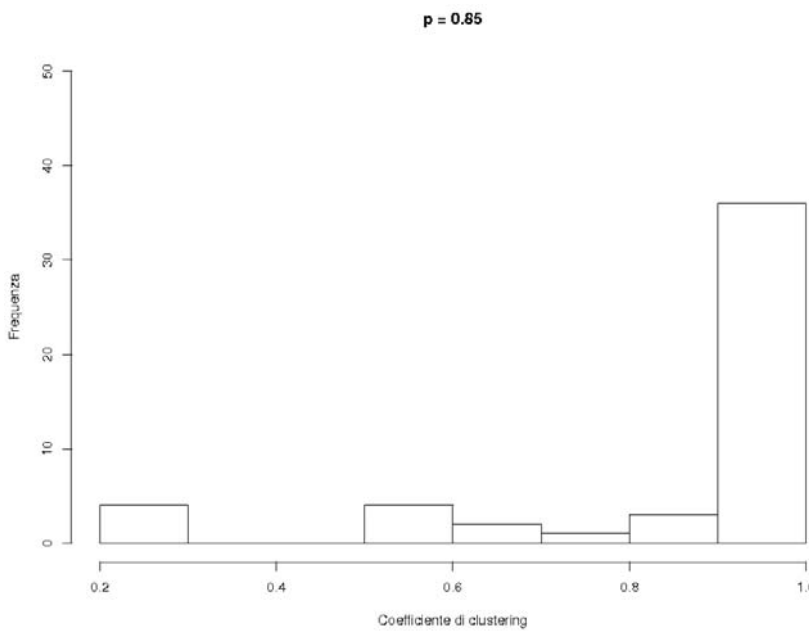



Figura 3: Distribuzione dei valori di coefficiente di clustering pesato relativo alla matrice delle distanze tra gli attrattori in un campione di 100 reti con N = 70, K = 3 e p = 0.85.

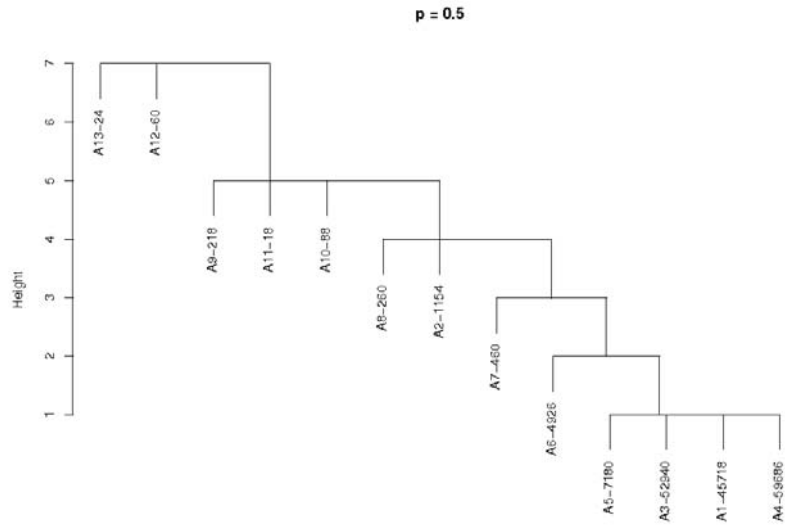



Figura 4: Dendrogramma relativo alla matrice delle distanze degli attrattori in un campione di 100 reti con N = 70, K = 3 e p = 0.5.

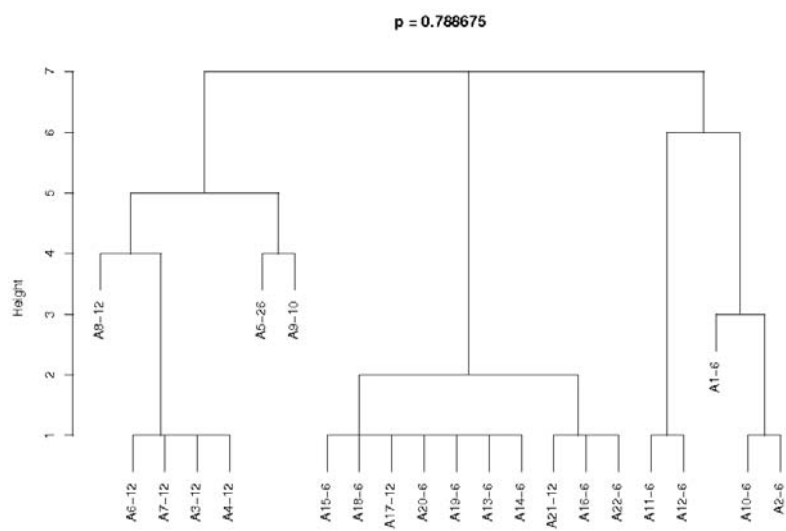



Figura 5: Dendrogramma relativo alla matrice delle distanze degli attrattori in un campione di 100 reti con N = 70, K = 3 e p = 0.788675.

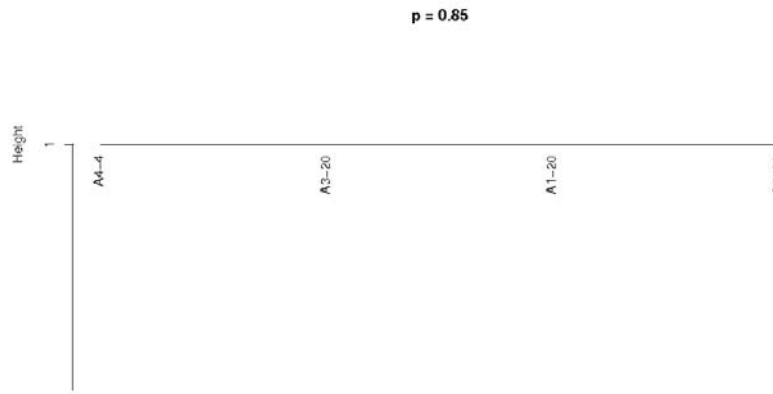



Figura 6: Dendrogramma relativo alla matrice delle distanze degli attrattori in un campione di 100 reti con N = 70, K = 3 e p = 0.85.

## *6. Conclusioni*

In questo contributo sono stati presentati i risultati di un'analisi sperimentale sulla distribuzione delle distanze degli attrattori di RBC. Si è osservato che la tendenza a formare cluster con attrattori a distanza molto piccola è molto marcata nelle reti ordinate e critiche, mentre le reti caotiche presentano un paesaggio meno organizzato e più sparso. Estensioni di questo lavoro prevedono la comparazione dei risultati qui presentati con quelli analoghi ottenuti nello spazio degli stati e l'analisi delle proprietà del landscape degli attrattori.

## *Bibliografia*

1. Kauffman, S.A.: The Origins of Order: Self-organization and Selection in Evolution. Oxford University Press, New York (1993)
2. Derrida, B., Pomeau, Y.: Random networks of automata: a simple annealed approximation. Europhys. Lett., vol. 1, (1986)
3. Aldana, M., Balleza, E., Kauffman, S.A., Resendiz, O.: Robustness and evolvability in genetic regulatory networks. J. of Theor. Biol., n. 245 (2007)
4. Zhang, B., Horvath, S.: A General Framework for Weighted Gene Co-expression Network Analysis. Stat. Appl. in Gen. and Mol. Biol., vol. 4, n.1 (2005)

5. Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Comp. Surv., vol. 31, n. 3 (1999)
6. Benedettini, S.: The Boolean Network Toolkit, http://booleannetwork.sourceforge.net (2009)

# *Un modello 3D di robotica evolutiva per lo sviluppo di controller autonomi per robot volanti*

**Fabio Ruini**
*Adaptive Behaviour and Cognition Research Group, Centre for Robotics and Neural Systems, University of Plymouth,* Drake Circus, PL4 8AA, Plymouth, United Kingdom
Tel. +44 (0)1752 586288, Fax +44 (0)1752 586200
fabio.ruini@plymouth.ac.uk

**Angelo Cangelosi**
*Adaptive Behaviour and Cognition Research Group, Centre for Robotics and Neural Systems, University of Plymouth,* Drake Circus, PL4 8AA, Plymouth, United Kingdom
Tel. +44 (0)1752 586288, Fax +44 (0)1752 586200
acangelosi@plymouth.ac.uk

## *1. Introduzione*

Il presente lavoro descrive alcuni risultati preliminari ottenuti dagli autori nell'estensione di un modello simulativo da loro precedentemente sviluppato (Ruini, Cangelosi & Zetule, 2009) (Ruini & Cangelosi, 2009). Il modello in questione è mirato all'evoluzione di controller autonomi, basati su reti neurali, per squadre di MAVs (Micro-unmanned Aerial Vehicles). L'approccio utilizzato per lo sviluppo di tali controller si basa su una combinazione di robotica evolutiva (Nolfi & Floreano, 2000) e sistemi multi-agente (Wooldridge, 2009).

## *2. Panoramica del modello 2D*

Il modello simulativo originario del quale trattiamo qui un'estensione si sviluppa in un ambiente bidimensionale. Al suo interno è prevista la presenza alternata di "squadre" composte da 4 velivoli ciascuna, impegnate nello svolgimento di un'operazione comune. All'inizio di

ogni epoca di test, un target viene dislocato in una certa posizione all'interno di questo ambiente. Nei vari scenari, basandosi su un mix di informazioni di carattere "globale" ed altre raccolte localmente, i MAVs, guidati da un controller neurale "embedded", devono riuscire a navigare fino al target e, una volta giunti in sua prossimita', mettere in atto una determinata operazione (rappresentata dall'attivazione di un particolare neurone booleano di output).

Il modello in questione presenta alcune caratteristiche innovative per quanto riguarda le simulazioni di robotica evolutiva. In particolare è presente una componente legata al moto degli agenti che raramente si trova negli esperimenti in letteratura. Nello scenario delineato qui sopra i MAVs devono infatti mantenere una velocità costante per tutta la durata della loro vita e non possono mai entrare in contatto con alcun ostacolo (sia esso un edificio, un altro velivolo o uno dei confini dell'ambiente), pena l'immediato fallimento del test nel quale sono impegnati. Questa condizione li costringe ad una pianificazione estremamente accurata (e "conservativa") di ciascun movimento: compito tanto più problematico se si considera il fatto che essi non sono in possesso di alcuna mappa, né predefinita né costruita in maniera dinamica, dell'ambiente di riferimento. Anche il modo in cui i MAVs devono operare una volta raggiunto il target rende il compito che essi devono svolgere particolarmente impegnativo. Essi non possono infatti attivare il succitato neurone booleano più di una volta, dato che il suo utilizzo provoca l'immediata "morte" dell'agente.

Durante il processo evolutivo, nonostante le costrizioni di cui sopra, accanto alle capacità standard di navigazione sono emersi in maniera relativamente semplice comportamenti di più alto livello. Sono stati studiati nel dettaglio quattro scenari: (1) ambiente sgombro da ostacoli; (2) ambiente con ostacoli (disposti in maniera tale da ricreare un'area urbana ispirata al Canary Wharf di Londra); (3) target in movimento, in grado di avvertire l'avvicinarsi di un MAV e tentare quindi di allontanarsi da esso; (4) target fisso (a) o in movimento (b) ed azione finale che richiede cooperazione tra i membri del team (due MAVs devono approcciare il target ed attivare il loro neurone booleano di output in rapida successione). In tutti questi casi, il processo evolutivo ha condotto a controller in grado di svolgere il compito richiesto, seppur con perfomance diverse a seconda del livello di difficoltà del compito. Nel dettaglio, la percentuale complessiva di test completati con successo per gli individui dell'ultima generazione è stata rispettivamente del 93.46% nello scenario 1, dell'87.18% nello scenario 2, del 76.4% (calcolato come media tra 5 simulazioni dove il target si muoveva ad altrettante dif-

ferenti velocità) nello scenario 3, del 72.3% nello scenario 4a ed infine del 49.6% nello scenario 4b.

## *3. Il passaggio al 3D*

Il passaggio ad un modello 3D presuppone l'aggiunta di due gradi di libertà (DOFs) agli agenti rispetto a quelli disponibili nel simulatore 2D (dove i MAVs potevano soltanto ruotare il loro corpo in senso orario o anti-orario, basandosi di fatto su un singolo DOF). Se consideriamo un sistema di riferimento ortogonale incentrato nel centro di massa dei velivoli simulati e che si muove con essi, le rotazioni che i MAVs possono effettuare sono tre: yaw, pitch e roll. Nel dettaglio, lo yaw è una rotazione dell'aereo lungo la sua asse verticale, mentre pitch e roll si riferiscono rispettivamente alle rotazioni che avvengono lungo l'asse che collega le due ali, e quelle attorno all'asse che va dalla coda al muso (vedi Figura 1).

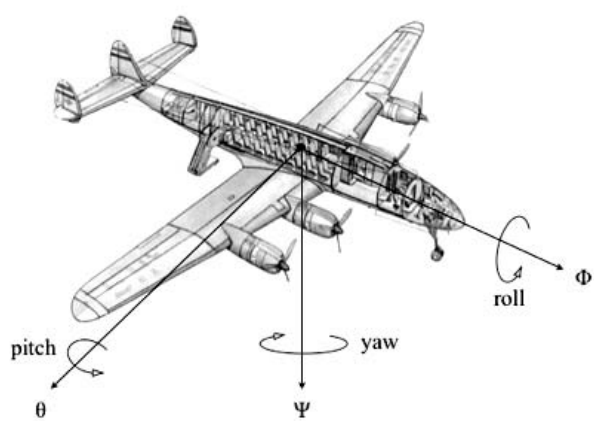



Figura 1. I gradi di libertà solitamente associati ad un velivolo ad ali fisse

Il controller di ciascun MAV è implementato mediante una rete neurale feed-forward. Le informazioni che essa riceve in input sono: (1) la distanza dal target, (2) l'angolo orizzontale tra target e MAV relativo alla direzione in cui quest'ultimo sta puntando, (3) la differenza di altitudine tra target e velivolo, l'angolazione corrente di roll (4) e pitch (5)

dell'aereo. In output sono presenti neuroni continui che modificano in maniera incrementale la rotazione del MAV lungo le tre assi, oltre al neurone booleano che implementa la "azione finale" che i velivoli devono mettere in atto una volta raggiunto il target.

L'evoluzione verso reti neurali in grado di esibire il comportamento desiderato è garantita dall'utilizzo di un algoritmo genetico (Mitchell, 2000). Una popolazione iniziale di 100 controller è creata con pesi sinaptici e bias assegnati casualmente nel range [-10.0; +10.0]. Ciascun controller è assegnato in maniera univoca ad un intero team di MAVs: ogni velivolo appartenente a tale squadra sarà pertanto guidato dalla medesima rete neurale. I team sono formati da 4 MAVs e vengono testati per 12 epoche (con il target dislocato in altrettante posizioni diverse) con i velivoli inizialmente distribuiti in prossimità dei quattro angoli dell'ambiente e dotati di una certa riserva di energia (la quale decresce con il passare del tempo, provocando la caduta del MAV in caso di esaurimento). L'ambiente di riferimento è sgombro da ostacoli ed il setup corrisponde pertanto allo scenario 1 del simulatore 2D. Una volta che tutti i team sono stati messi alla prova e valutati attraverso una funzione di fitness[1], i migliori 10 vengono selezionati per la riproduzione. Ciascuno di essi genera 9 copie del suo genoma, sui cui pesi e bias è applicata con probabilità 0.15 una mutazione casuale che ne modifica il valore di un ammontare casuale nel range [-0.2; +0.2]. L'operatore di elitismo è applicato al fine di preservare il miglior controller di ciascuna generazione. 9 nuovi controller, con pesi e bias assegnati in maniera casuale, vengono creati ad ogni generazione. L'intero processo viene iterato per 10,000 generazioni e ripetuto per 20 volte, al fine di limitare il più possibile gli effetti dovuti al caso.

Varie architetture di rete sono state testate (si veda il riassunto in Tabella I). Si tratta di diverse topologie di reti feed-forward, che possono essere dotate o meno di uno strato intermedio e che ricevono in input informazioni in valore continuo o discretizzate. Le architetture differiscono anche per come il roll è stato implementato (esso può essere assente, generato come effetto secondario dello yaw oppure utilizzato in maniera indipendente), nonché per la presenza di memorie di breve termine, implementate sotto forma di reti di Elman (Elman, 1990) o di Jordan (Jordan, 1986) a seconda dei casi.

[1] La funzione di fitness è data dalla semplice formula: $f = -\alpha + \beta$, dove $\alpha$ è la distanza, calcolata come media per i 12 test, tra il target ed il velivolo del team che più vi è arrivato vicino nel momento in cui ha attivato il suo neurone booleano di output; $\beta$ è invece l'ammontare medio di energia rimasta nei serbatoi dei MAVs che hanno portato a termine con successo la missione.

Tabella I. Le architetture di rete utilizzate nel simulatore 3D

| Arch. NN | Input | Strato nascosto | Roll | Memoria |
|---|---|---|---|---|
| 1 | D | No | No | No |
| 2 | C | No | No | No |
| 3 | D | No | Con yaw | No |
| 4 | C | No | Con yaw | No |
| 5 | D | No | Indipendente | No |
| 6 | C | No | Indipendente | No |
| 7 | D | Sì | Con yaw | No |
| 8 | C | Sì | Con yaw | No |
| 9 | D | Sì | Indipendente | No |
| 10 | C | Sì | Indipendente | No |
| 11 | D | No | Con yaw | Jordan |
| 12 | C | No | Con yaw | Jordan |
| 13 | D | Sì | Con yaw | Elman |
| 14 | C | Sì | Con yaw | Elman |
| 15 | D | No | Indipendente | Jordan |
| 16 | C | No | Indipendente | Jordan |
| 17 | D | Sì | Indipendente | Elman |
| 18 | C | Sì | Indipendente | Elman |

## *4. Risultati preliminari in 3D e confronto con il 2D*

I risultati ottenuti con il nuovo simulatore sono riassunti in Tabella II.

Così come era atteso, le architetture 1 e 2 sono quelle che hanno in assoluto generato i risultati migliori. Si tratta però di due simulazioni di benchmark, che, non implementando in nessun modo il roll, generano movimenti dei MAVs non realistici[2]. Aspetto interessante è che i controller sembrano funzionare meglio quando possono gestire in maniera indipendente il roll, rispetto a quando questo è legato allo yaw. Il tutto nonostante la gestione diretta di questa componente aumenti il livello di complessità della rete e le dimensioni dello spazio delle soluzioni. Le reti multi-strato dotate di uno strato intermedio hanno fatto

[2] Il simulatore 3D, allo stato attuale, non si basa su alcun engine fisico, ma l'idea alla base è quella di far sì che i MAVs esibiscano comunque movimenti che non siano in palese contrasto con le principali leggi della fisica.

registrare performance generalmente migliori rispetto a quelle dove lo strato di input era direttamente connesso a quello di output. L'introduzione di ricorrenze, a parte aver rallentato in maniera significativa il processo evolutivo, non ha condotto ad un miglioramento delle prestazioni. Più ambiguo è invece il ruolo giocato dalla discretizzazione dell'input sensoriale, che in alcuni casi ha determinato vantaggi rispetto all'utilizzo di input continui, mentre ha condotto in altre circostanze a deterioramenti delle performance.

Tabella II. Risultati ottenuti con il simulatore 3D per le varie architetture di rete

| Arch. NN | Fitness media | Fitness massima | Percentuale successi |
|---|---|---|---|
| 1 | 126.5897 | 461.002 | 75.49 |
| 2 | 168.2282 | 454.7155 | 76.65 |
| 3 | -130.3004 | 345.6638 | 43.63 |
| 4 | -35.2221 | 407.9116 | 54.93 |
| 5 | 53.7617 | 406.5243 | 60.26 |
| 6 | 5.4618 | 391.213 | 57.49 |
| 7 | -82.8986 | 360.323 | 47.54 |
| 8 | 49.8237 | 438.7521 | 63.96 |
| 9 | 119.3603 | 444.415 | 67.55 |
| 10 | 53.28 | 420.1875 | 60.28 |
| 11 | -221.9481 | 232.8541 | 20.86 |
| 12 | -160.9756 | 256.3188 | 24.72 |
| 13 | -258.8055 | 113.8254 | 12.95 |
| 14 | -265.2192 | 130.6172 | 15.5 |
| 15 | -42.8824 | 311.3557 | 35.81 |
| 16 | -18.4841 | 346.6911 | 45.3 |
| 17 | -92.2134 | 291.5339 | 33.46 |
| 18 | -85.5089 | 296.8809 | 34.21 |

## *5. Multi-threading*

Il maggior livello di complessità derivante dall'utilizzo di un simulatore 3D ha stimolato l'investigazione sugli effetti prodotti dal ricorso a procedure di programmazione multi-threading. Il codice del motore evolutivo (che è stato scorporato dalla parte grafica in maniera tale da

alleggerire il carico di lavoro richiesto durante l'evoluzione) è stato pertanto modificato in maniera tale da poter sfruttare tutta la potenza di calcolo disponibile sui calcolatori in uso[3], distribuendo i team di MAVs da testare sui vari core a disposizione.

I benefici che questa modifica ha apportato, per quanto positivi, sono stati inferiori rispetto alle nostre attese. Quello che ci aspettavamo era che il passare dall'utilizzo di un singolo core a 8 in contemporanea avrebbe comportato un corrispondente incremento di almeno il 600% della velocità di esecuzione del codice (o comunque un valore leggermente inferiore rispetto al teorico +700%, per via di qualche inevitabile overhead). In realtà, le analisi effettuate hanno mostrato come il miglioramento di performance abbia luogo soltanto con una magnitudo più ridotta. Sono stati condotti 10 esperimenti, nei quali sono state evolute 50 generazioni per l'architettura neurale 9, sia in single che in multi-threading. Nel primo caso, portare a termine il test ha richiesto una media di 150,354.8 msec; nel secondo caso, invece, 27,765.1 msec. L'incremento di velocità nel passaggio al multi-threading è sicuramente significativo (+441.52%), ma ben lontano rispetto al teorico +700% ottenibile.

## *6. Conclusioni*

In questo articolo è stata descritta la prima parte di un lavoro mirato all'estensione del pre-esistente simulatore 2D in un nuovo modello 3D, nonché descritta quella che è stata l'esperienza degli autori nell'utilizzo di metodologie di programmazione multi-threading. Il livello di realismo del precedente modello è stato incrementato e questi sviluppi costituiscono pertanto un passo aggiuntivo lungo la strada verso il possibile trasferimento dei controller sviluppati in simulazione su piattaforme robotiche reali.

Malgrado un generale decremento delle performance dei MAVs nello svolgere compiti che nell'analogo simulatore 2D venivano portati a termine in maniera relativamente semplice, il processo evolutivo ha comunque condotto a risultati positivi anche per quanto riguarda il nuovo modello. Il peggioramento delle prestazioni era un risultato atte-

[3] Le simulazioni sono state eseguite su quattro Apple Xserve, ottenuti da Apple nel contesto del programma ARTS (http://www.apple.com/uk/education/hed/arts/). Ciascuna di queste macchine è mossa da un doppio processore Intel Xeon Quad-Core a 2.8GHz. Il multi-threading è stato ottenuto sfruttando le apposite funzioni offerte dalle librerie Qt (http://qt.nokia.com).

so in quanto l'aggiunta di nuovi gradi di libertà al simulatore aumenta notevolmente le dimensioni dello spazio delle soluzioni che l'algoritmo evolutivo deve esplorare alla ricerca di un'appropriata configurazione di pesi sinaptici e bias per i controller.

I risultati presentati in relazione al multi-threading non sono sufficientemente solidi e tantomeno generalizzabili al punto da permettere di trarre da essi conclusioni definitive. Questi suggeriscono però che il ricorso a metodologie di programmazione parallela debba essere pianificato con attenzione e criterio, in quanto non necessariamente conduce verso i miglioramenti attesi nella velocità di esecuzione delle simulazioni. Nel caso specifico, data la relativa velocità nell'effettuare la valutazione di un singolo team, una possibile soluzione potrebbe essere quella di modificare il codice in maniera tale da distribuire su ciascun core disponibile la valutazione di un maggior numero di squadre, riducendo in questo modo l'inevitabile impatto dell'overhead. In aggiunta alla replica degli esperimenti effettuati con il simulatore 2D, parte dei lavori futuri sarà dedicata ad approfondire questo aspetto.

Disclaimer

Efforts sponsored by the Air Force Office of Scientific Research, Air Office Material Command, USAF, under grant number FA8655-07-1-3075.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon.

## *Bibliografia*

1. Ruini, F., Cangelosi, A., Zetule, F.: Individual and Cooperative Tasks performed by Autonomous MAV Teams driven by Embodied Neural Network Controllers. In: Proceedings of the 2009 International Joint Conference on Neural Networks, pp. 2717-2724 (2009)

2. Ruini, F., Cangelosi, A.: Extending the Evolutionary Robotics Approach to Flying Machines: an Application to MAV Teams. Neural Networks, 22, 812-821 (2009)
3. Nolfi, S., Floreano, D.: Evolutionary Robotics. MIT Press, Cambridge, MA (2000)
4. Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley & Sons, Hoboken, NJ (2009)
5. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA (1998)
6. Jordan, M.I.: Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 531-546 (1986)
7. Elman, J.L.: Finding Structure in Time. Cognitive Science, 14, 179-211 (1990).

# *BestBot*
# *Un gioco di vita artificiale*

**Massimiliano Schembri**
*Istituto di Scienze e Tecnologie della Cognizione – CNR - Roma*
Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: +39 06-44595201
massimiliano.schembri@istc.cnr.it

**Andrea Di Ferdinando**
*Dipartimento di Scienze Relazionali, Università di Napoli "Federico II"*;
Via Porta di Massa, 1 80113 Napoli,
Tel. +39 0812535466, Fax +39 0812535634
andrea.diferdinando@istc.cnr.it

**Alberto Venditti**
*Istituto di Scienze e Tecnologie della Cognizione – CNR - Roma*
Via S. Martino della Battaglia 44, 00185 Roma (RM)
Tel.: +39 06-44595201
alberto.venditti@istc.cnr.it

## *1. Introduzione*

BestBot rappresenta una nuova tipologia di giochi in cui, a differenza di quelli tradizionali, il giocatore non controlla direttamente un'entità (sia esso un pilota, un calciatore, un guerriero, un animale, e così via), ma svolge piuttosto il ruolo di insegnante nei confronti di tale entità.

Non si tratta quindi di far valere la propria prontezza di riflessi ed abilità manuale, o la propria bravura nel risolvere degli enigmi, quanto piuttosto di elaborare una propria strategia di addestramento, e di poter controllare la bontà dei propri "metodi di allenamento" osservando i progressi fatti dall'"allievo". Inoltre, è possibile organizzare una sfida con gli allievi di altri giocatori, ed assistere quindi da spettatori alle loro "prestazioni sul campo", ovviamente non mancando di incitarli.

Nel caso di BestBot, poi, il giocatore per plasmare l'allievo secondo i propri criteri, deve saper sfruttare a proprio vantaggio i principi del-

l'evoluzione naturale. L'esperienza di gioco è dunque unica, ed al contempo permette di imparare divertendosi alcuni concetti di tipo scientifico [2, 4, 5].

## *2. Il gioco*

In BestBot dobbiamo addestrare un robot a percorrere, nel minor tempo possibile, una serie di percorsi. Il robot è provvisto di sensori (infrarossi e telecamera) e di motori (ruote) e grazie alle informazioni che ricava dall'ambiente deve imparare a muoversi nel modo più veloce possibile, evitando gli ostacoli e scegliendo la strada più breve.

Per addestrare il robot, possiamo far uso dei cosiddetti Algoritmi Genetici, ossia modelli computazioni dell'evoluzione naturale. In pratica, partendo da robot che si muovono in maniera del tutto casuale, possiamo "dirigere" l'evoluzione verso la creazione di robot sempre più intelligenti, scegliendo accuratamente chi far sopravvivere e chi invece eliminare. La trasmissione del patrimonio genetico dai genitori ai figli fa sì che i geni migliori tendano a diffondersi maggiormente, o meglio i geni che secondo noi sono i migliori. Proprio nella capacità di discernere i migliori robot (e quindi i corrispondenti geni) sta infatti la nostra bravura come insegnanti. Al termine dell'evoluzione, così, avremo tra le mani il robot che più degli altri ci soddisfa.

Sia il robot sia i percorsi vengono simulati al computer, ma in realtà, una volta ottenuto un robot che riteniamo valido, possiamo trasferire la sua conoscenza in un robot reale, fisico, e testarlo su una serie di percorsi anch'essi reali.

Inoltre, possiamo sfidare altri giocatori ed assistere alla gara tra i propri robot. Al termine della sfida, viene loro assegnato un punteggio in base al risultato finale. La classifica dei migliori robot è consultabile sul sito del gioco. Il migliore robot in assoluto è il BestBot.

## *3. La piattaforma*

BestBot è un sistema integrato software/hardware, formato da tre componenti fondamentali:

– una parte client, che verrà rilasciata per la prima volta all'interno del Festival di Città della Scienza di Napoli "Futuro Remoto 2009",

fruibile attraverso tutti i più noti browser web e scaricabile anche sul proprio computer.
- un robot reale, con sensoristica Lego Mindstorm NXT © per poter testare il proprio robot allevato nel mondo reale [3].
- un sito web, dove è possibile sfidare altri giocatori e assistere in diretta alla gara tra i corrispondenti robot, nonché consultare la classifica aggiornata dei migliori robot e le statistiche riguardanti il nostro.

### *3.1 Il Client*

Il programma Client sarà usufruibile direttamente sul sito del gioco mediante l'installazione di un plugin, nonché scaricabile dal sito di Futuro Remoto 2009 (www.futuroremoto.it) e installabile sul proprio computer. Tramite il programma, il giocatore avrà a disposizione, tramite una semplice interfaccia, tutte le possibilità di interazione con il gioco.

La principale di esse è quella di creare un robot e permette al giocatore di poter ottenere, come un vero allevatore di robot, il proprio "campione", sfruttando come detto dei speciali Algoritmi Genetici. Per ottenere un buon robot, il giocatore può creare anche diversi ambienti, modificando il terreno, aggiungendo ostacoli e posizionando eventuali concorrenti. Il tutto per fornire al robot quella capacità di generalizzare che lo può rendere più forte nelle sfide con gli avversari.
Il robot è controllato da una rete neurale, e gli Algoritmi Genetici agiscono sia sull'architettura sia sui pesi di essa.

Quando il giocatore è soddisfatto del suo "pupillo", può osservarne le capacità in un robot reale, oppure sfidare il pupillo di un altro giocatore.

### *3.2 Il robot reale*

Come detto, è possibile trasferire la conoscenza dei robot simulati, ossia la loro rete neurale, in un robot reale. Per realizzarlo, si è scelto di utilizzare sensoristica Lego Mindstorm NXT © in una scocca costruita da ricercatori del Laboratorio di Robotica Autonoma e Vita Artificiale dell'Istituto di Scienze e Tecnologie della Cognizione del CNR.

La compatibile tra i dati simulati e quelli reali è garantita dal fatto che BestBot è compatibile con il simulatore EvoRobot [1]. Più specifi-

catamente, è stato creato un binding tra EvoRobot ed il motore del gioco, ed in particolare con la libreria Unity3D sulla quale BestBot si appoggia.

### *3.3 Il Web*

La parte online del gioco permette una visione complessiva dei giocatori che partecipano al campionato con i loro robot addestrati. Ogni robot addestrato ha infatti un certo punteggio in base alle gare vinte con i concorrenti ed è per questo in una certa posizione in classifica. Ogni giocatore, accedendo con i suoi dati nell'area web, può sfidare qualsiasi robot di un altro giocatore per raggiungere la vetta della classifica ed aggiudicarsi la posizione di BestBot.

Per attuare la sfida il giocatore può utilizzare il suo Client. la sfida avviene in un ambiente scelto casualmente, ed è per questo motivo che in fase di addestramento il giocatore deve prestare particolare attenzione alla scelta degli ambienti dove testare il proprio robot, affinché esso sia in grado di generalizzare gli "insegnamenti" del giocatore, e non rimanere vincolato agli ambienti di testing.

Alla fine della gara il risultato aggiornerà la classifica dei due robot sfidanti, nonché le relative statistiche. Chissà che il nostro robot non sia diventato il BestBot.

## *4. Conclusioni*

La costruzione della piattaforma BestBot pone le basi per un nuovo trend di sviluppo e di ricerca. Le applicazioni di vita artificiale stanno infatti riscuotendo un sempre crescente interesse nelle comunità esterne alla ricerca. Questo gioco verrà inaugurato all'interno del Festival di Futuro Remoto 2009 a Città della Scienza di Napoli, proprio per la volontà di integrare la ricerca e la tecnologia con la divulgazione e l'apertura verso comunità esterne alla vita artificiale.

## *Bibliografia*

1. Nolfi S. and Floreano D. [2000]. Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press/Bradford Books

2. Miglino, O., Gigliotta, O., Ponticorvo, M., Lund, H.H. (in press). Human Breeders for Evolving Robots. Artificial Life and Robotics Journal.
3. Miglino, O., Ponticorvo, M., Rega, A., Di Martino, B. (2008) Robotics Exhibits for Science Centres. Some Prototypes. In A. Gottscheber, D. Obdrzalek, J.P. Mendoza, J.-D.Dessimoz, S. Enderle (Eds.). Proceedings of the EUROBOT Conference 2008 Heidelberg.
4. Miglino, O., Gigliotta, O., Ponticorvo, M., Nolfi, S. (2008) Breedbot: an evolutionary robotics application in digital content. The Electronic Library, 26,(3) 363- 373.
5. Miglino, O., Gigliotta, O., Ponticorvo, M. (2008.) Allevare robot: Una nuova prospettiva. Sistemi Intelligenti, 2, 249-257.

# *Cellule artificiali: dall'attuale quadro teorico-sperimentale al loro uso come robot molecolari*

**Pasquale Stano**
*Dipartimento di Biologia,*
*Università RomaTre*
Viale Guglielmo Marconi, 446
00146 Roma, Italia
Tel. +39 0657336433, Fax +39 0657336321
stano@uniroma3.it

## *1. Introduzione*

La vecchia tradizione organicistica in biologia è stata in gran parte soppiantata dall'avvento della genetica moderna e della biologia molecolare, che hanno dominato il panorama negli ultimi 40 anni. L'apice di questa impresa, cioè la mappatura del genoma umano, corrisponde paradossalmente alla crisi di questo approccio riduzionista. Stephen J. Gould nel 2001 osservò l'esigenza di un nuovo spostamento di paradigma in biologia: "*The collapse of the doctrine of one gene for one protein, and one direction of causal flow from basic codes to elaborate totality, marks the failure of reductionism for the complex system that we call biology*" (New York Times, 19/02/2001).

Negli ultimi anni, fortunatamente, c'è stata un rinnovato e fiorente in-teresse verso la visione sistemica degli organismi, evidente nello sviluppo della biologia dei sistemi e - più recentemente - della biologia sintetica. Entrambe le discipline guardano agli organismi nella loro totalità, avvalendosi delle conoscenze generate in passato da approcci riduzionistici, ma con un nuovo e caratteristico Zeitgeist, che è specificamente orientato verso punto di vista un sistemico (globale).

È all'interno di queste nuove correnti che la ricerca su "cellule sintetiche" trova il suo posto ottimale, anche se un'analisi epistemologica rivelerebbe le sue salde radici in campi come la cibernetica, l'autoorganizzazione, la chimica prebiotica e le origini della vita.

La moderna ricerca sulla vita cellulare "minimale" inizia cercando di rispondere alla vecchia domanda "che cosa è la vita?". Si chiede inoltre come sia stato possibile - per un sistema chimico primitivo - evolvere per livelli crescenti di complessità, dal caos destrutturato primordiale fino alla vera e propria vita cellulare.

La ricostruzione storica del percorso d'evoluzione molecolare è piuttosto complicata. Ci si chiede se sia possibile trovare un percorso plausibile che conduca da molecole semplici a più complesse, poi a protocellule e, infine, a cellule viventi. Questa strada, definibile come approccio "bottom-up", è piuttosto difficile da percorrere, e nonostante alcuni progressi nella chimica prebiotica, restano aperte diverse problematiche.

Una possibile alternativa è rappresentata dall'approccio "top-down", che si concentra sulla sintesi (costruzione) di forme di vita minima a partire dall'analisi delle moderne cellule. L'approccio "top-down" è sperimentalmente possibile, e può essere classificato come uno degli approcci costruttivisti alla conoscenza scientifica. L'atto di costruire è visto come un modo di ottenere informazioni sulla natura stessa della cellula. Sono possibili diverse implementazioni della vita minimale: protocellule o cellule primitive, le cellule minimali, i bioreattori, i robot molecolari (soft-robots).

Un corollario importante da menzionare si origina dal collegamento tra questo approccio di "vita minima" con la teoria autopoietica di Maturana e Varala, che enfatizzano la conservazione dell'organizzazione strutturale e funzionale di una cellula, che avviene per assimilazione e trasformazione di composti disponibili nell'ambiente, al fine di costruire le parti molecolari che la compongono. Tali parti sono in grado di autoassemblarsi e di autoorganizzarsi in strutture e processi, che a loro volta sono responsabili alla trasformazione di molecole semplici nelle parti del sistema, e così via ... Il sistema definisce le sue dinamiche attraverso una logica circolare, in cui le strutture e funzioni sono inevitabilmente connesse, e in continuo rigenerarsi a spese dell'ambiente.

Ne consegue che i tentativi moderni di costruire la vita si riducono alla sintesi di cellule minime autopoietiche.

Molteplici studi dedicati alle cellule artificiali sono stati effettuati "in silico". Il termine "cellula artificiale" è, infatti, tradizionalmente legato all'analisi modellistica computerizzata di tali strutture minime. L'aspetto innovativo sviluppatosi negli ultimi anni è invece legato al fatto che i progressi tecnologici hanno reso possibile la costruzione di (semplici) sistemi chimici-biochimici in laboratorio. Essi consistono di solito di oggetti simili a cellule (ad esempio, vescicole lipidiche) di di-

mensione micrometrica o sub-micrometrica, costituiti da migliaia di molecole, dotate di un confine semi-permeabile e in grado di interagire con l'ambiente.

## *2. Lo stato dell'arte nelle cellule minimali semi-sintetiche*

Anche se non molto conosciuto, il lavoro avviato da Thomas M.S. Chang circa quarant'anni fa, consistente nell'incapsulamento di estratti cellulari all'interno di vescicole artificiali di cellulosa o nylon, può essere considerato come il primo vero tentativo di progettare (e costruire) una cellula sintetica [1].

L'attuale strategia sperimentale considera le cellule sintetiche come costituite da due sottosistemi principali, interagenti tra loro: il compartimento (tipicamente vescicole lipidiche) e il contenuto (biomacromolecole tipicamente funzionali come il DNA, l'RNA, enzimi, ribosomi, PNA, ribozimi, peptidi catalitici, ecc.)

Per prima cosa, occorre avere il controllo della formazione del compartimento, in termini di tecniche applicate per la sua costruzione e in termini di composizione chimica. Inoltre, è necessario il controllo sperimentale dell'intrappolamento. Negli ultimi 15 anni sono stati effettuati notevoli progressi riguardo il controllo di questi due importanti processi. La tappa successiva è la costruzione di vescicole con un ricco contenuto di soluti, diversi in dimensione, proprietà, funzioni. Ciò è stato realizzato solo di recente (dopo il 1999 [2]) mediante l'intrappolamento dell'intero set di molecole necessarie alla trascrizione e la traduzione di geni all'interno di vescicole lipidiche. I primi lavori furono dedicati a sistemi più semplici [3]. Oggi, il caso più ben studiato è l'espressione di proteine funzionali all'interno di vescicole lipidiche. Le ragioni di questa scelta sono molteplici: (i) dal punto di vista teorico, la sintesi delle proteine è una delle funzioni chiave per una cellula vivente, comprendente circa al 40% del genoma minimo, e quindi la sua realizzazione in cellule sintetiche farà in modo che ulteriori obiettivi possano essere potenzialmente raggiunti, (ii) gli estratti cellulari e i kit ricostituiti (come PURESYSTEM [4]) sono facilmente accessibili e ben caratterizzati, (iii) la reazione è sufficientemente complessa per essere usata come paradigma del metabolismo cellulare, e può servire per modellizzare il comportamento stocastico e gli effetti di crowding nelle reazioni microcompartimentalizzate.

Una recente panoramica sui dati sperimentali disponibili la si può trovare in letteratura [5]. In particolare, grazie alla convergenza della

tecnologia dei liposomi e delle tecniche cell-free (in vitro), è possibile sintetizzare una o più proteine funzionali all'interno di liposomi, con dimensioni da 200 nm a 100 micron. Ciò significa che le funzioni metaboliche, sensoriali e strutturali possono essere codificate in elementi genetici all'interno di cellule sintetiche costruite in laboratorio. Recenti studi si occupano di analizzare quantitativamente la sintesi proteica tenendo conto in modo esplicito la diversità della popolazione di cellule artificiali [6].

### *3. Prossimi obiettivi*

È chiaro che la ricerca sulle cellule artificiali è ad un punto di svolta. I necessari avanzamenti possono essere classificati in due modi: (1) funzionali, (2) tecnici. Alla prima classe appartengono tutte le sfide per l'implementazione, in cellule sintetiche, di funzioni che vanno oltre la sintesi proteica. Esempi di tali obiettivi non ancora raggiunti sono:

- l'attuazione di un semplice metabolismo in vescicole;
- l'attuazione di un dispositivo di trasduzione di energia (ad esempio, l'energia radiante in energia chimica) o di un sistema per trasformare l'energia chimica (gradiente chemiosmotico, o chimico) in energia biochimica, ecc.;
- riproduzione di tutti i componenti interni delle cellule e della membrana;
- l'accoppiamento e sincronizzazione tra i due processi sopra menzionati;
- lo sviluppo di una funzione sensoriale rispetto all'ambiente e, conseguentemente, l'attivazione di sistemi di risposta;
- la comunicazione tra le cellule sintetiche - o tra le celle sintetiche e naturali [7];
- il movimento

Tra i progressi tecnici, è necessario sviluppare un metodo efficace per l'assemblaggio di cellule, che dia origine ad una popolazione omogenea di cellule sintetiche in termini di dimensioni e di contenuto. Recenti progressi nei dispositivi microfluidici possono rivelarsi risolutivi. A partire dal successo applicativo di tecniche come la compartimentazione in vitro per lo screening di librerie geniche [8], alcune nuovi studi puntano verso la produzione di cellule sintetiche in apparati a microcanali [9,10].

Grazie a questi metodi, potrebbe essere possibile creare uno standard per le cellule sintetiche, simile a quello descritto nel Registro di Standard Biologia parti del MIT [11].

### *4. Cellule sintetiche e robot molecolari*

Le cellule sintetiche possono essere considerate come robots su scala supramolecolare, e costituiti da materiale "soft", vale a dire proteine, DNA, lipidi, ecc. La loro peculiarità è di esistere in soluzione acquosa. Inoltre, in contrasto con i robot meccanici, i cui pezzi sono stati assemblati dagli esseri umani, i robot molecolari sono composti da parti che si auto-assemblano e si auto-organizzano.

Tra le applicazioni più semplici delle cellule sintetiche, troviamo, naturalmente, lo sviluppo di complessi sistemi di veicolazione di farmaci – l'evoluzione del moderno impiego dei liposomi in medicina. Un'intera via metabolica potrebbe essere veicolata mediante soft robots. Le Duc immagina un'interazione stimolo-risposta tra il robot molecolare e le cellule nell'organismo, reso possibile grazie ai circuiti genetici interni della cellula sintetica [12].

Nella diagnostica, le cellule sintetiche possono funzionare come biosensori, per esempio per realizzare saggi enzimatici accoppiati, con diversi enzimi co-intrappolati (o sintetizzati) all'interno del comparto stesso [13]. Il biosensore può essere utilizzato anche per lo sviluppo di microarrays di liposomi. Per mezzo della ricostruzione di semplici circuiti genetici AND, OR, NOT [14] all'interno di liposomi, sarà forse possibile utilizzare nanorobot per una sorta di calcolo. Alcuni risultati indicano che è possibile pensare al "bioputing" con popolazioni batteriche [15].

È inoltre prevedibile che con l'aumento della nostra capacità di manipolare le cellule sintetiche, si costruiranno architetture sempre più complesse (ad esempio in struttura multi-vescicolare), o clusters di cellule sintetiche all'interno di gocce d'acqua in emulsione. Tali costrutti possono avere un ruolo come biosensori o come strumento d'analisi per cellule naturali. Certamente possono essere concepite molte più applicazioni, per ora siamo solo limitati dalla nostra immaginazione.

## *Bibliografia*


1. Ming, T.S.C.: Artificial Cells. Charles C. Thomas Publisher, Springfield (1972)
2. Oberholzer, T., Nierhaus, K.H., Luisi, P.L.: Protein Expression in Liposomes. Biochim. Biophys. Res. Comm. 261, 238--241 (1999)
3. Oberholzer, T., Albrizio, T., Luisi, P.L.: Polymerase Chain Reaction in Liposomes. Chem. & Biol. 2, 677--682 (1995)
4. Shimizu, Y., Inoue, A., Tomari, Y., Suzuki, T., Yokogawa, T., Nishikawa, K., Ueda, T.: Cell-free Translation Reconstituted with Purified Components. Nat. Biotechnol. 19, 751--755 (2001)
5. Chiarabelli, C., Stano, P., Luisi, P.L.: Chemical Approaches to Synthetic Biology. Curr. Opin. Biotech. in press (2009)
6. Hosoda, K., Sunami, T., Kazuta, Y., Matsuura, T., Suzuki, H., Yomo, T.: Quantitative Study of the Structure of Multilamellar Giant Liposomes as a Container of Protein Synthesis Reaction. Langmuir 24, 13540--13548 (2008).
7. Gardner, P.M., Winzer, K., Davis, B.G.: Sugar Synthesis in a Protocellular Model leads to a Cell Signalling Response in Bacteria. Nat. Chem. 1, 377--383 (2009.
8. Taly, V., Kelly, B.T., Griffiths, A.D.: Droplets as Microreactors for High-Throughput Biology. ChemBioChem 8, 263--272 (2007)
9. Sugira, S., Kuroiwa, T., Kagota, T., Nakajima, M., Sato, S., Mukataka, S., Walde, P., Ichikawa, S.: Novel Method for Obtaining Homogeneous Giant Vesicles from a Monodisperse Water-in-Oil Emulsion Prepared with a Microfluidic Device. Langmuir 24, 4581--4588 (2008).
10. Ota, S., Yoshizawa, S., Takeuchi, S.: Microfluidic Formation of Monodisperse, Cell-Sized, and Unilamellar Vesicles. Angew. Chem. Int. Ed. 48, 6533--6537 (2009)
11. Massachusetts Institute for Technology, www.parts.mit.edu
12. Zhang, Y., Ruder, W.C., Le Duc, P.R.: Artificial cells: Building Bioinspired Systems using Small-Scale Biology. Trends Biotech. 26, 14--20 (2008)
13. Pohorille, A., Deamer, D.: Artificial Cells: Prospects for Biotechnology. Trends Biotech. 20, 123--128 (2002)
14. Sprinzak, D., Elowitz, M.B.: Reconstruction of Genetic Circuits. Nature 438, doi:10.1038/nature04335 (2005)
15. Baumgardner, J., Acker, K., Adefuye, O., Crowley, S.T., DeLoache, W., O Dickson, J., Heard, L., Martens, A.T., Morton, N., Ritter, M., Shoecraft, A., Treece, J., Unziker, M., Valencia, A., Waters, M., Campbell, M., Heyer, L.J., Poet, J.L., Eckdahl, T.T.: Solving a Hamiltonian Path Problem with a Bacterial Computer. J. Biol. Engineer 3, 11 (2009)

# *I curatori*

**Orazio Miglino** è professore ordinario di psicologia generale, responsabile scientifico del Laboratorio per lo Studio dei Sistemi Cognitivi Naturali e Artificiali e coordinatore del dottorato in scienze psicologiche e pedagogiche presso l'Università degli Studi di Napoli Federico II. I suoi interessi di ricerca riguardano principalmente il campo delle Scienze Cognitive e della Vita Artificiale.

**Michela Ponticorvo** è assegnista di ricerca e membro del Laboratorio per lo Studio dei Sistemi Cognitivi Naturali e Artificiali presso l'Università degli Studi di Napoli Federico II. La sua attività di ricerca è focalizzata, attualmente, sulla costruzione di modelli dell'orientamento spaziale nell'ambito delle scienze cognitive e della vita artificiale.

**Angelo Rega** è laureato in psicologia e dottorando di ricerca presso il dipartimento di scienze relazionali "G. Iacono" dell' l'Università degli Studi di Napoli Federico II. Gli interessi di ricerca riguardano lo sviluppo di tecnologie per la riabilitazione e l'apprendimento di soggetti normali e patologici.

**Franco Rubinacci** è assegnista di ricerca presso l'Istituto di Scienze e Tecnologie della Cognizione del CNR di Roma e membro del Laboratorio per lo Studio dei Sistemi Cognitivi Naturali e Artificiali dell'Università degli Studi di Napoli Federico II. L'attività di ricerca riguarda prevalentemente il campo delle Tecnologie Cognitive e della loro applicazione in contesti formativi.

# *Elenco Autori Wivace 2009*

In questo volume trovate raccolti i lavori presentati al workshop **WIVACE 2009** tenutosi presso l'Università di Napoli "Federico II" nel novembre del 2009 e organizzato dal Laboratorio per lo studio dei sistemi cognitivi naturali e artificiali (www.nac.unina.it).
La comunità interdisciplinare di Vita Artificiale e Computazione Evolutiva condivide l'obiettivo scientifico della costruzione di artefatti che abbiano alcune proprietà del vivente, dalla chimica, alla biologia, alla genetica, passando per la psicologia e utilizzando gli strumenti della matematica, della fisica e dell'informatica per il duplice scopo, indicato da Langton, di contribuire alla comprensione dei principi su cui si fonda la vita e di applicare tali principi alla soluzione di problemi scientifici e tecnologici.