Università degli Studi di Bari

Facoltà di Scienze MM. FF. NN.

Dipartimento Interateneo di Fisica "*M. Merlin*"

Tesi di Laurea Triennale

L'EVOLUZIONE SIMULATA:

VITA ARTIFICIALE E

ALGORITMI GENETICI

Relatore:

Prof. *Savino Longo*

Laureando:

Marco Di Gennaro

Anno Accademico 2008/2009

INTRODUZIONE

Con la presente Tesi di Laurea in Fisica il laureando vuole proporre una breve panoramica su un fervido ambiente di studio teorico che offre una valida metodologia di risoluzione per svariati problemi.
Tale metodo adotta particolari algoritmi , detti genetici (AG) in quanto emulano metodi di ottimizzazione biologici ed in particolare genetici.
Nel nuovo campo vengono mescolati argomenti appartenenti a discipline diverse (biologia, fisica e informatica) con un approccio squisitamente computazionale.
È necessario quindi introdurre dei concetti generali relativi a queste materie, come le nozioni di selezione naturale e codice genetico, che stanno alla base delle iterazioni degli AG e sono indispensabili per lo sviluppo logico del procedimento. La maggior parte dei termini utilizzati vengono mutuati dalle scienze naturali.
L'ampliamento delle nozioni, sia teoriche che informatiche, e la diversificazione delle metodologie hanno dato vita a differenti rami, nati dalla stessa branca e oramai divenuti materie autonome.
Le applicazioni possibili sono innumerevoli: dalla progettazione delle proteine, alla ricerca di equazioni che permettano di predire l'andamento di un mercato finanziario, alla soluzione di qualsiasi problema di tipo combinatoriale.
I vantaggi consistono nello svincolare la risoluzione dei problemi da ragionamenti umani a volte troppo angusti e non sempre efficaci.

1.L'ANALISI DELL'EVOLUZIONE

Il cervello animale, le cellule e il sistema immunitario sono meccanismi di eccezionale complessità e specificità. Le scienze naturali hanno dimostrato che questi meccanismi sono il risultato di un numero elevatissimo di iterazioni delle stesse semplici regole.
Il punto di partenza del nostro lavoro è la comprensione di tali regole, riunite sotto il complesso dell'idea di evoluzione naturale.
Successivamente si cerca di emularle artificialmente, per piegarle a migliorare la tecnica di risoluzione dei nostri quesiti.
Il problema che ci proponiamo di risolvere è il nostro ambiente, si creano dunque una popolazione di diverse soluzioni candidate alla risoluzione che "abitano" questo ambiente.
Si associa alla popolazione un codice genetico, si stabiliscono i metodi per l'approvvigionamento delle risorse naturali e per per stabilire il *fitness* (stato di salute) degli individui.
Si utilizzano degli operatori di riproduzione e di mutazione genetica che operano sulle soluzioni ispirandosi alla naturale variabilità genetica ed alla selezione naturale per far evolvere la popolazione e lasciare che la specie più *forte* si imponga sulle altre.
Dunque il concetto di fondo è che le soluzioni si sviluppano ed evolvono sotto la pressione naturale alla competizione e all'automiglioramento, giungendo autonomamente al risultato migliore per il problema, senza che il programmatore agisca dall'esterno.

1.1 – EVOLUZIONE: SELEZIONE NATURALE E GENETCA

Così come per il concetto di atomo, il concetto di selezione

naturale sembrava bizzarro e assolutamente rivoluzionario quando Charles Darwin lo introdusse nel 1859 con il libro *L'origine delle specie.* Oggi è largamente accettata l'idea che è la Natura stessa che porta le diverse specie a competere tra loro per l'approvvigionamento delle risorse naturali le quali, essendo limitate, non sono sufficienti a garantire la sopravvivenza di tutti gli esseri.
I principi fondamentali su cui si basa la selezione naturale sono i seguenti:

- il principio della variabilità dei caratteri tra gli individui di una popolazione;
- il principio dell'adattamento, secondo il quale gli individui che presentano caratteri vantaggiosi ai fini della sopravvivenza e della riproduzione sono i più *adatti* all'ambiente;
- il principio dell'ereditarietà dei caratteri tra genitori e figli.

Darwin proponeva per l'ultimo punto una teoria puramente speculativa detta pangenesi, poi confutata, secondo la quale i genitori trasmettono alla progenie informazioni acquisite in vita.
Fu Gregor Mendel a fornire la base teorica alla questione dell'ereditarietà dei caratteri, affermando che i caratteri tramandati sono indipendenti dalla vita passata dei genitori.
La genetica Mendeliana individua lo strumento del quale la natura si serve per tramandare i caratteri, l'informazione genetica, la cui possibilità di mutazione fornisce anche una valida spiegazione all'elevatissimo grado di diversità biologica presente sulla Terra.
Gli individui di una stessa specie[1] si differenziano l'uno dall'altro per caratteristiche genetiche o del genotipo e fenotipiche (cioè morfologiche e funzionali, frutto dell'interazione del genotipo con l'ambiente).
I genitori, riproducendosi, mescolano tra loro il proprio corredo genetico e lo trasmettono alla progenie.
Qualora si verifichi una mutazione genetica, questa viene conservata e diffonde velocemente nelle generazioni successive solo se risulta positiva. La specie quindi evolve. Altrimenti, se il nuovo carattere non risulta migliorativo viene velocemente perso.
Pertanto nell'ambito della diversità genetica delle popolazioni, si ha un progressivo (e cumulativo) aumento della frequenza degli individui più

[1]La definizione di specie più comune è quella zoologica, secondo la quale due individui appartengono alla stessa specie se, riproducendosi, sono in grado di generare una prole illimitatamente fertile. Più in generale la specie è l'unità tassonomica fondamentale della biologia.

adatti.
In altre parole, è l'ambiente a selezionare le mutazioni secondo il criterio detto di vantaggiosità: i geni forieri di vantaggio adattativo vengono trasmessi e con il susseguirsi delle generazioni si ha una progressiva affermazione dei geni utili a discapito dei geni inutili o dannosi.

1.2 – CENNI STORICI

Probabilmente i padri dell'informatica – Alan Turing, John Von Neumann, Norbert Wiener, per citarne alcuni – erano motivati dalla comune spinta di creare, infondendola nei programmi, l'intelligenza, intesa come capacità di adattamento e capacità di *problem solving*.
La biologia era probabilmente vista come la scienza madre alla quale ispirarsi per conseguire tale scopo.
John Von Neumann interveniva al simposio Hixon, tenuto da Linus Pauling a Pasadena nei tardi anni 40, con un discorso intitolato "La teoria e la logica generali degli automi", definendo *"automa: una qualsiasi macchina il cui comportamento proceda passo per passo combinando le informazioni ricevute dall'ambiente con la propria programmazione"* e affermando che alla fine si sarebbe potuto dimostrare come *"gli organismi naturali si comportino in maniera analoga".*
Non sorprende dunque che, in parallelo agli utilizzi militari, i calcolatori furono da subito impiegati per costruire modelli del cervello animale, cercando di emulare l'apprendimento e la capacità di comunicare.
Sempre a Von Neumann si devono i primi passi nello studio della logica per l'auto-riproduzione delle macchine. Il suo scopo era quello di ottenere il carattere dell'universalità, cioè la possibilità di costruire un qualsiasi pattern finito di celle. È suo infatti il primo progetto sulla vita artificiale universale, un automatismo cellulare self-reproducing in grado di evolvere autonomamente mediante autoriproduzione ed evoluzione di strutture adattative complesse (cfr 1.3).
Norbert Wiener dal canto suo applicò la teoria dell'informazione e l'analisi dei processi auto-regolamentati allo studio dei sistemi viventi. Nel prossimo paragrafo queste tematiche verranno approfondite.
Il conio dell'espressione "Vita Artificiale" si deve all'informatico Christopher Langton il quale ha tenuto la prima "Conferenza

Internazionale sulla Sintesi e Simulazione dei Sistemi Viventi" (anche nota come Artificial Life I) presso il Laboratorio Nazionale di Los Alamos, nel 1987.
Gli studiosi di vita artificiale non hanno avuto nessuna organizzazione fino ai tardi anni 80, e hanno spesso lavorato isolati, inconsapevoli di altri impegnati negli stessi studi. Nei rari casi in cui si sono occupati dell'intelligenza, questi ricercatori si sono concentrati maggiormente sulla natura induttiva dei comportamenti emergenti.

1.3 - IL CAMPO DELLA VITA ARTIFICIALE.

La vita artificiale (artificial life, alife o a-life) ha come scopo lo studio e la realizzazione di sistemi artificiali che mostrano un comportamento *quasi vivo* comparato alla vita “classica” fondata sulla chimica del carbonio.
La comune definizione di “vita” non considera le simulazioni computazionali come viventi, né tanto meno questi costituiscono parte del processo di evoluzione di un qualsivoglia ecosistema.
Una definizione precisa è difficile da fornire, poiché si andrebbe a sovrapporre alle definizioni di altre discipline, che si occupano dello stesso campo ma con metodi e finalità differenti.
Vengono spesso distinte tre branche per l'*a-life*, dai confini non sempre ben delineati tra loro, sebbene altre distinzioni siano possibili:

- La vita artificiale forte (cioè basata su un'ipotesi più forte rispetto alle altre) si basa sul fatto che "la vita è un processo che può essere astratto da ogni mezzo particolare" (John Von Neumann). Si cerca di produrre implementazioni hardware di sistemi life-like.

- La vita artificiale debole crea simulazioni o semplici costruzioni digitali che presentano comportamenti life-like. I suoi sostenitori non sostengono la possibilità di generare un processo vitale al di fuori di una soluzione biochimica basata sul carbonio e tentano invece di imitare i processi vitali per capire l'apparire dei singoli fenomeni. Il metodo più usuale è l'uso di modelli basati su agenti, che normalmente forniscono una soluzione di tipo minimale.

- La vita artificiale wet che sintetizza sistemi viventi fuori dalle sostanze biochimiche.

L'*a-life* affronta un vasto range di fenomeni, dalla riproduzione delle molecole all'evoluzione di intere specie e dinamiche sociali e culturali che si presentano quando gli agenti evolvono imparando o comunicando l'un l'altro.
I metodi utilizzati sono la simulazione e i modelli computazionali.
Data la vastità dell'argomento risulta opportuno fare riferimento alle altre materie che si occupano degli stessi argomenti senza scendere troppo in particolari che sarebbero fuorvianti rispetto allo scopo della tesi.
Tra questi ci sono la teoria dell'informazione, l'intelligenza artificiale e la scienza cognitiva. Vengono inoltre accennati i concetti di automa cellulari, macchina di Turing, network neurale e di comportamento adattivo, La vita artificiale è utilizzata anche nell'arte visuale e nella musica.

COMPORTAMENTO ADATTATIVO

Taluni agenti autonomi sono in grado di modificare in maniera autosufficiente il loro comportamento, sostituendo ad un'eventuale condotta anti-costruttiva o addirittura dannosa una più produttiva.
L'attenzione degli studi è focalizzata sul meccanismo mediante il quale i singoli agenti possono coordinare la percezione e l'azione al fine di sopravvivere per un lungo periodo in un ambiente generalmente dinamico e incerto.
Molti programmi computazionali richiedono tale proprietà, si pensi ad esempio ad interfacce uomo-macchina sfruttate da utenti diversi.
Molti degli agenti autonomi sviluppati usano networks neurali artificiali come controllori per coordinare percezione e azione.

SCIENZA COGNITIVA

Si occupa del funzionamento di un qualsiasi sistema (sia esso naturale o artificiale), capace di percepire, pensare, ricordare, e di comunicare tali informazioni ad un altro sistema. Si analizza come questi sistemi agiscono nel mondo adattandosi ai suoi cambiamenti, o adattando il mondo a se stesso tramite la creazione di artefatti.

TEORIA DELL'INFORMAZIONE

È una branca della matematica che si occupa della quantificazione dell'informazione, sviluppata in principio per trovare i limiti per la compressibilità e l'affidabilità dei dati.
Una chiave di misura è l'entropia di informazione che è normalmente espressa dal numero medio di bit richiesto per lo stoccaggio o la comunicazione dell'informazione. Intuitivamente tale nozione di entropia è legata all'incertezza che si incontra di fronte ad una variabile random. Quindi l'entropia connessa al lancio di una moneta è minore rispetto a quella legata al lancio di un dado a sei facce.
Alcune comuni applicazioni di questi argomenti si trovano nella compressione di dati (ZIP files, MP3s), ma risultano utili in tutti i sistemi che comunicano tra loro, dai cromosomi ai computers connessi ad internet.

RETI NEURALI

Una rete neurale artificiale è un modello matematico/informatico di calcolo costituito da un gruppo di punti, detti neuroni, connessi tra loro, in grado di scambiare informazioni. Ogni nodo che riceve un'informazione la elabora e la trasmette ai nodi successivi. Si realizza così un sistema adattivo che cambia la sua struttura basandosi su informazioni esterne o interne che scorrono attraverso la rete durante la fase di apprendimento. Le strutture che si formano sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione. Tali modelli sono usati per sviluppare i cervelli degli agenti. Fondamentale per simulazioni di sistemi dinamici in grado di apprendere. L'integrazione a più livelli tra la capacità di apprendimento e di adattamento, e l'evoluzione è centrale nello sviluppo di teorie di organismi complessi.

INTELLIGENZA ARTIFICIALE (IA)

Lo studio della vita artificiale si sovrappone in modo significativo allo studio dell'intelligenza artificiale, i due campi sono tuttavia differenti sia per la loro storia sia per il loro approccio. Entrambi simulano fenomeni naturali.
La ricerca organizzata sull'*IA* è cominciata presto nella storia degli elaboratori digitali, ed era caratterizzata da un forte ottimismo. È stata spesso contraddistinta da un approccio analitico (top down), basato su complicati insiemi di regole, con un controllore centralizzato che prende decisioni basate su tutti gli aspetti dello stato globale. Sono le sue decisioni che dirigono gli aspetti dell'intero sistema.
In seguito ci si è resi conto che tali regole erano troppo complesse, del resto molti sistemi con comportamento autonomo risultano avere lo stesso tipo di comportamento, con un network centrale relativamente semplice e vari agenti di basso livello interagenti.
Ogni decisione dell'agente è basata solo sul proprio ambiente locale, ed è questo l'approccio seguito dall'*AL*, il bottom up, cioè sistemi di regole molto semplici, scritte dall'uomo e applicate parallelamente, in grado però di ottenere comportamenti complessi.

AUTOMI CELLULARI

La ricerca incentrata sugli AC aveva come scopo originale quello di offrire i requisiti logici indispensabili allo sviluppo dell'auto-riproduzione delle macchine. In seguito sono stati largamente utilizzati per studiare mediante simulazioni fenomeni che dipendono esclusivamente da leggi locali. Esempi di fenomeni di questo tipo sono il comportamento fisico dei gas perfetti, l'evoluzione di una popolazione, il movimento dei filamenti di RNA in una soluzione.
Un AC è un sistema complesso costituito da una sequenza finita di unità che si trovano in uno stato definito, capaci scambiare informazioni reciprocamente con i propri vicini. Ad ogni iterazione tutte le unità cambiano stato contemporaneamente, in base a regole che dipendono dal proprio stato e da quello dei propri vicini (entro una certa distanza). Il sistema quindi evolve nel tempo in maniera diversa a seconda delle regole imposte dal programmatore.
Il caso più semplice da presentare è quello unidimensionale, in cui gli unici stati consentiti sono 1 o 0. Considerando un set di tre celle adiacenti ci sono $2^3 = 8$ configurazioni possibili per un vicinato. Quindi possiamo imporre un totale di $2^8 = 256$ regole.

Il nome dell'automa è il numero decimale che, in notazione binaria, ci fornisce la tabella delle regole, con elencati gli 8 vicinati possibili elencati[2]. A titolo di esempio vengono di seguito mostrati i risultati che si ottengono in base a diversi pattern di evoluzione. La regola 30 in particolare è stata introdotta da Stephen Wolfram nel 1983 e utilizzata nel suo programma *Mathematica* per la generazione di numeri pseudo-casuali.
Il pattern seguito è il seguente ($(30)_{10}=(11110)_2$):

Pattern corrente	111	110	101	100	011	010	001	000
Nuovo stato per la cella centrale	0	0	0	1	1	1	1	0

Dunque graficamente, si ha:

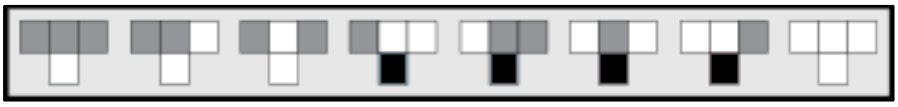

Fig 1.1 – Regola 30. Ogni pattern (in alto) genera un valore diverso (in basso)

L'AC viene inizializzato sempre mettendo una cella nera al centro del diagramma. Dopo molte iterazioni, spostando ogni volta tutto il corpo di righe precedentemente ottenuto di una posizione più in alto, si ottiene una figura del genere, che mostra l'evoluzione temporale dell'AC:

[2] Tale convenzione è dovuta a Wolfram

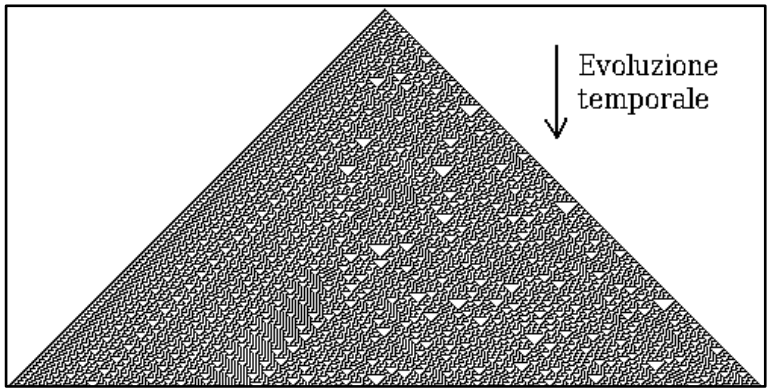

Fig 1.2 – AC (regola 30)

Un altro risultato si ha seguendo la *regola 110:*

Pattern corrente	111	110	101	100	011	010	001	000
Nuovo stato per la cella centrale	0	1	1	0	1	1	1	0

Notiamo che anche in questo caso $(110)_{10}=(1101110)_2$.
La quale da origine al seguente AC

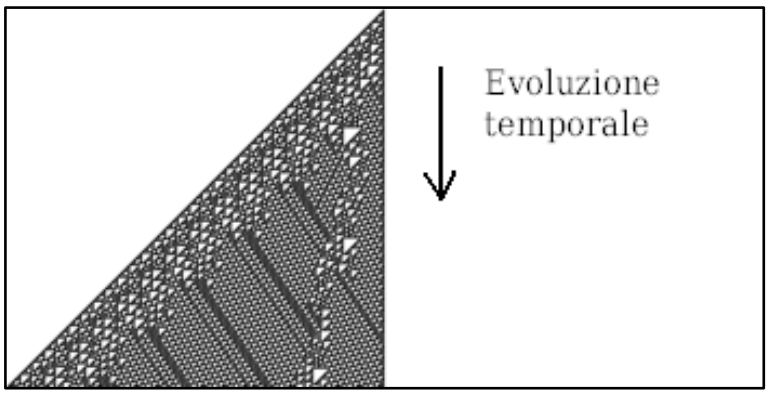

Fig. 1.3 – AC (regola 110)

Tra le 256 AC elementari, la regola 110 (insieme ad altre regole, quali la 124) risulta essere asimmetrica, o direzionale. È facile dimostrarlo se teniamo conto che le due cellule a destra di quella centrale iniziano

con l'essere bianche (*00), e hanno solo due possibilità, cioè che la cella centrale sia bianca (000) o nera (100), ma i ogni caso si ha:

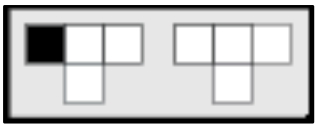

Fig 1.4 – Particolare della regola 110

Nel 2000 Matthew Cook ha verificato una congettura introdotta da Stephen Wolfram nel 1985, dimostrando che l'AC 110 è una macchina di Turing completa, o universale, cioè una macchina di Turing capace di simulare le evoluzioni di una qualsiasi altra macchina di Turing. Una macchina di Turing (MdT) è un dispositivo ideale che può trovarsi in stati ben determinati, opera su stringhe mediante lettura e scrittura in base a regole semplici e costituisce un modello di calcolo. La macchina si evolve nel tempo e ad ogni istante. Si presume che essa abbia la portata computazionale massima. La MdT come modello di calcolo è stato introdotta nel 1936 da Alan Turing per dare risposta all'*Entscheidungsproblem* (problema di decisione).
Per le sue caratteristiche, il modello della MdT è un efficace strumento teorico che viene largamente usato nella teoria della calcolabilità e nello studio della complessità degli algoritmi. Per definire in modo formalmente preciso la nozione di algoritmo oggi preferenzialmente si sceglie di ricondurlo alle elaborazioni effettuabili con macchine di Turing.
La regola 110 è presumibilmente la macchina di Turing completa più semplice.

Nel corso degli anni, la ricerca sulla vita artificiale è servita anche come ombrello temporaneo per diverse tecniche che non sarebbero state accettate in altri campi.
L'*a-life* ricorre all'uso estensivo di programmi e simulazioni computazionali, che includono il calcolo evolutivo (Algoritmi evolutivi, Algoritmi genetici, Programmazione genetica, swarm intelligence e Ant Colony Optimization[3]), la chimica artificiale, i modelli basati su agenti

[3]. Rispettivamente Intelligenza di sciame e Ottimizzazione basata sui formicai

e gli automi cellulari. Spesso queste tecniche sono state considerate come sotto-campi della vita artificiale fino a che i rispettivi campi non sono cresciuti abbastanza da guadagnarsi una posizione autonoma, e quindi conferenze specifiche.
La ricerca sull'*a-life* è risultata un punto di incontro per molti campi di ricerca più tradizionali come fisica, matematica, filosofia, informatica e biologia, ma anche linguistica, psicologia, scienze etnoantropologiche, e sociologia, discipline per le quali approcci computazionali e teorici, giudicati normalmente inusuali e controversi, possono essere discussi con importanti risultati.

2. ALGORITMI GENETICI

2.1 GLI ALGORITMI GENETICI (AG)

Seguendo regole semplici e identiche per le diverse specie, la selezione naturale è riuscita a fornire la straordinaria diversità biologica che osserviamo nella biosfera.
Agli AG si richiede di comportarsi in maniera simile, cioè si richiede che trovino delle soluzioni a problemi con condizioni mutevoli seguendo una serie finita di passaggi standard.
Se consideriamo ad esempio il sistema immunitario dei mammiferi,

questo è la soluzione migliore che finora si sia avuta per difendere i corpo da virus e batteri. È in questo senso che la ricerca computazionale si ispira all'evoluzione.
Non si ha una definizione rigorosa per gli AG, sono metodi euristici di ricerca ed ottimizzazione, ispirati al principio della selezione naturale di Charles Darwin. Vengono utilizzati sia come algoritmi adattivi, sia come modelli computazionali di sistemi evolutivi naturali.
Piuttosto con AG si intende una serie di metodi che in comune hanno il fatto di agire su una popolazione di cromosomi che vengono selezionati in base al loro fitness e vengono fatti riprodurre e mutare, ottenendo l'evoluzione delle soluzioni.
Ogni cromosoma è composto di geni, cioè da una stringa di bit, ciascuno con più possibilità. Nel caso le possibilità siano due si parla di allele.
Ogni cromosoma può essere pensato come un punto in un spazio delle soluzioni candidate (la dimensione di questo spazio sarà 2^l con l numero dei bit). L'AG modifica le popolazioni di cromosomi, in base al valore della loro funzione di fitness, cioè il grado di risolvibilità del problema posto.
Banalmente, assegnata una funzione a valori reali f(x), le soluzioni candidate sono tutti i valori x del dominio di f. Questi possono essere codificati in stringhe di bit, per esempio scrivendoli in formato floating point. Il valore della funzione di fitness per la soluzione x può essere proprio il valore assunto dalla funzione in x. La soluzione migliore è quel particolare numero x_0 che massimizza f(x).
La funzione di fitness, o idoneità, pur essendo imposta dall'esterno riflette sempre il problema stesso, risulta quindi interna ad esso. Può essere scelta ad esempio come la longevità o la fertilità dell'individuo.
Gli algoritmi genetici sono applicabili alla risoluzione di un'ampia varietà di problemi d'ottimizzazione non indicati per gli algoritmi classici, compresi quelli in cui la funzione obiettivo è discontinua, non derivabile, stocastica, o fortemente non lineare.
Consideriamo il seguente problema: una popolazione di individui appartenenti a specie diverse si trova in un ambiente con una certa disponibilità di cibo. Ogni individuo cerca di sopravvivere, nutrendosi come meglio può e riproducendosi. Ognuno dei membri della popolazione possiede un certo corredo genetico, due individui in questo caso appartengono alla stessa specie se possono incrociarsi generando una prole illimitatamente feconda. Viene definita la funzione di fitness, cioè un indice della salute di cui gode un individuo, intesa come energia disponibile per muoversi alla ricerca di cibo.
All'inizializzazione dell'algoritmo si ha la generazione di una popolazione in maniera casuale. A ogni iterazione l'algoritmo prevede

la misura del valore di fitness di tutti gli individui. Vengono selezionati quindi i membri con la salute migliore e si procede a farli riprodurre.
La riproduzione si ha essenzialmente grazie al meccanismo di crossing over, i cui parametri (numero e posizione dei geni scambiati) vengono stabiliti esternamente. I nuovi individui sostituiscono gli elementi precedenti.
Come già anticipato, la nuova generazione tende a conservare i geni migliori, i quali permettono l'accaparramento di cibo con un minore utilizzo di energia.
I meccanismi di mutazione genetica portano ad una variazione casuale nel genotipo di un individuo, che può o meno favorirlo. Solo le modifiche positive tendono a sopravvivere.
Finita la fase di evoluzione la popolazione delle soluzioni viene analizzata e vengono tenute solo le soluzioni che meglio risolvono il problema: gli individui con le qualità più adatte all'ambiente in cui si trovano hanno quindi maggiori possibilità di sopravvivere e riprodursi. Queste soluzioni subiranno una nuova fase di evoluzione e così via.
Alla fine ci si aspetta di trovare una popolazione di soluzioni che riescano a risolvere adeguatamente il problema posto. Non vi è modo di decidere a priori se l'algoritmo sarà effettivamente in grado di trovare una soluzione accettabile.

2.2 - OPERATORI DEGLI AG

Nella sua forma più semplice un AG necessita di almeno tre operatori:

SELEZIONE

Seleziona i cromosomi adatti alla riproduzione, quanto più è alta la funzione di fitness tanto più frequentemente esso sarà scelto per la riproduzione.

CROSSING OVER - Single point crossing over:

Tra le varie tecniche di crossing over, una delle più semplici è la *single*

point crossing over[4], nella quale l'operatore di selezione taglia le stringhe di codifica genetica degli elementi selezionati in un punto a caso con una probabilità p_c. Si creano così due teste e due code che si ricompongono incrociandosi.

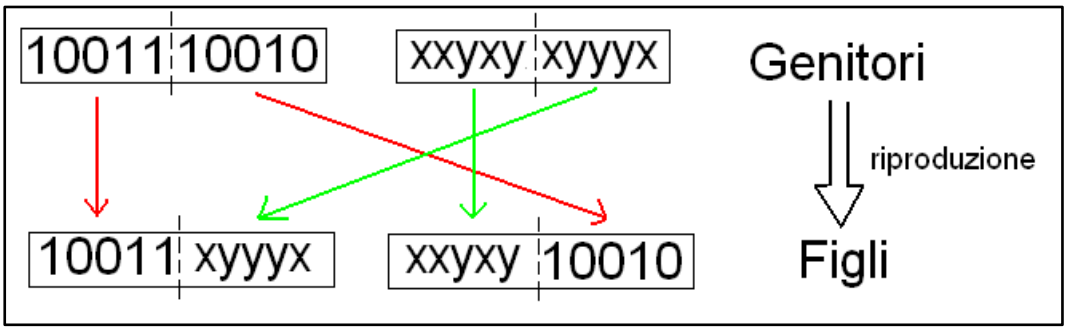

Fig 2.1 - crossing over di cromosomi

È una rozza imitazione del sistema di riproduzione fra due organismi aploidi[5].
Nel caso in cui il crossing over non viene applicato (è un avvenimento dal carattere comunque stocastico) i figli sono semplicemente le copie dei genitori.
I miglioramenti diventano apprezzabili solo dopo un certo numero di passi. Non è detto infatti che i cromosomi figli siano necessariamente più idonei all'ambiente dei genitori, dato che i geni *buoni* possono essere distrutti in una singola riproduzione, ma sono quelli che tendono a sopravvivere dopo molto tempo.

MUTAZIONE

L'operatore di mutazione agisce modificando casualmente alcune parti di un singolo gene, in particolare inverte i geni codificati in binario.
È importante evidenziare che, a prescindere dal fitness, tutti i cromosomi devono essere soggetti a mutazione con la stessa probabilità p_m (di valore generalmente basso).
Se ci si limitasse a far mutare solamente i geni deboli si andrebbe a finire in "buche" di ottimo locale, creando dei casi molto fortunati e

[4] Tale tecnica è dovuta a John H. Holland
[5] Organismi formati da un solo cromosoma

poco realistici.
Inoltre la mutazione non permette grossi spostamenti nello spazio delle soluzioni. Il crossing over al contrario fornisce la possibilità di mescolare geni forieri di proprietà isolate rispetto al resto perché c'è già stata l'operazione di selezione. L'*ensemble* statistico così analizzato alla ricerca di forme ancor migliori di evoluzione, oltre ad essere più esteso, risulta essere anche formato da elementi privilegiati, cioè con un fitness più alto rispetto alla media.

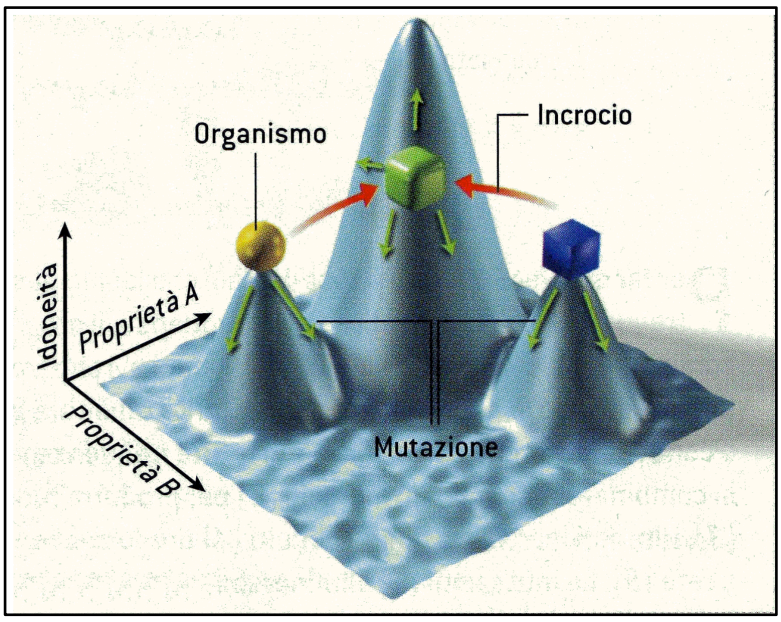

Fig 2.2 – Soluzioni candidate nello spazio delle soluzioni
(tratto da J.H.Koza, M.A.Kane e M.J.Streeter; "L'evoluzione delle invenzioni", *Le Scienze*, Marzo 2003

Riassumendo l'algoritmo evolve quindi attraverso i seguenti punti:

1. generazione di una popolazione iniziale casuale, formata da n cromosomi ad m bit;
2. calcolo del valore di fitness per tutti gli individui;
3. generazione nuova popolazione (ripete i seguenti passaggi finché non vengono creati n discendenti, uno per cromosoma):
   a) selezione di una coppia di genitori;
   b) esegue crossing over con probabilità p_c, e sostituisce ai genitori i due nuovi cromosomi. Se non avviene incrocio i due

cromosomi sono la copia esatta dei genitori;

c) muta i due discendenti in ogni gene con probabilità p_m;

4. sostituisce la nuova popolazione alla vecchia;
5. verifica che il criterio di arresto sia soddisfatto, altrimenti torna al punto 2.

Le differenti generazioni sono le varie iterazioni che il programma subisce. Normalmente si ha un numero di iterazioni dell'ordine di 500, prima di ottenere una coppia di cromosomi con un valore di fitness tale da soddisfare il criterio di arresto.

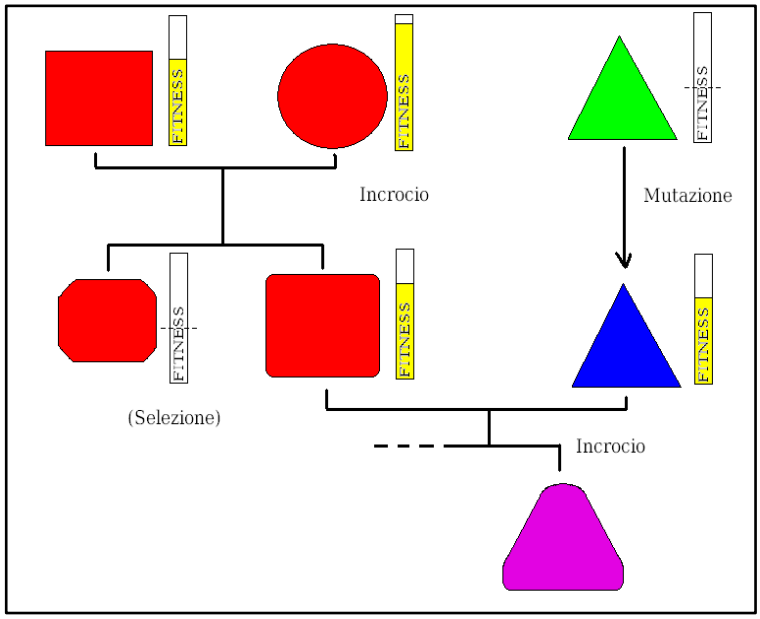

Fig 2.3 – Azione combinata degli operatori di Selezione, Incrocio e Mutazione

ESEMPIO

Si consideri il caso in cui gli organismi primordiali siano funzioni matematiche (algebriche, logaritmiche...) e che l'obiettivo sia quello di trovare la funzione il cui grafico sia quello che meglio approssimi una certa curva γ.
L'idoneità di una funzione f(x) è tanto maggiore quanto più questo requisito è soddisfatto. Potremmo considerare come funzione di fitness

il funzionale preso come segue:

$$\Phi(f) = \frac{1}{1+\int_a^b |f(x)-\gamma_x| dx} \qquad (2.1)$$

con γ_x valore che la curva assume in corrispondenza di x.
Φ(f) raggiunge il suo massimo valore (1) quando f(x) ricalca perfettamente γ.

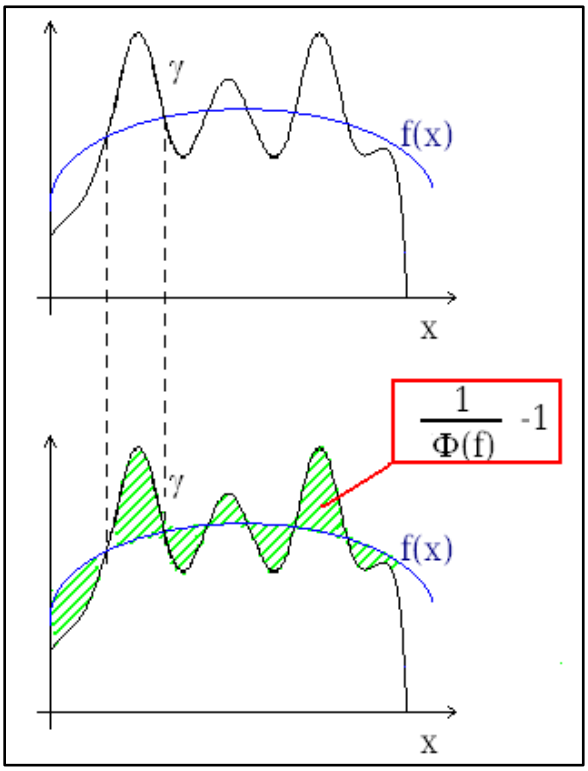

Fig 2.4 – Fitness della funzione f(x)

Le prime funzioni generate casualmente non riescono a riprodurre bene la curva data, ma esistono comunque delle differenze tra queste. La programmazione genetica tende a scartare le funzioni peggiori nella popolazione e applica gli operatori genetici alle migliori.

C'è da notare come non è detto che la soluzione al problema venga a trovarsi in forma esplicita già nella prima popolazione. È molto più frequente che le soluzioni candidate vengano create man mano che si procede (cfr 4.2).
Si pensi ad esempio alla differenza che corre tra un problema di ricerca di informazioni tipo: "qual'è il modo migliore per cercare un certo cognome in un vasto database?" e un problema tipo "risolvi il gioco dell'otto".
Mentre nel primo si tratta semplicemente di trovare un'informazione nel secondo ad ogni spostamento si apre un ventaglio di nuove possibilità da valutare singolarmente. Quindi la soluzione esatta è una lista di mosse. Alcune tecniche di IA sono più appropriate in questo caso perché permettono di valutare soluzioni parziali, sebbene non sono metodi sempre applicabili.

2.3 - COME FUNZIONANO GLI AG? IL TEOREMA DEI BLOCCHI

L'invenzione del concetto di AG e i primi passi allo sviluppo della teoria e alla ricerca sugli AG è opera di John .H.Holland. Egli, insieme ai suoi studenti della University of Michigan, ha perseguito lo scopo (ed ha avuto la geniale idea) di studiare le dinamiche generali che governano l'adattamento in natura e trasporle in analoghi informatici, piuttosto che cercare di progettare algoritmi per un problema specifico. Egli presentava l'algoritmo genetico come un'astrazione dell'evoluzione biologica, fornendo una valida base teorica.
La formulazione tradizionale della teoria sugli AG (1975) ipotizza che le soluzioni migliori tendano ad essere costituite da blocchi costitutivi "buoni", cioè da piccole sequenze di bit (non necessariamente vicini) che aumentano il valore della funzione di fitness del cromosoma.
Nel *teorema fondamentale degli algoritmi genetici (*o *teorema dei blocchi*) Holland dimostra che gli individui così costituiti tendono a crescere esponenzialmente nella popolazione attraverso il meccanismo dell'incrocio, assicurando così la convergenza dell'algoritmo genetico verso una soluzione ottimale.
L'algoritmo funziona individuando, favorendo e ricombinando tali blocchi in maniera parallela.
Viene qui presentato il teorema così come presentato da Holland, relativo cioè alla codifica binaria. Sono stati dimostrati teoremi

analoghi, con ipotesi simili ma che funzionano in casi più generali.
È necessario introdurre il concetto di *schema*, cioè un insieme di stringhe di una certa lunghezza formate da elementi fissi (dunque 1 o 0) e da elementi variabili, indicati con un asterisco (*), cioè si tratta di una particolare combinazione di geni che occupano posizioni precise all'interno di un cromosoma.
Ad esempio due esempi di schemi sono i seguenti:

$$H_a = 1 * * * * 0 ;$$

$$H_b = * * * * * 10 ;$$

cioè rispettivamente l'insieme di tutte le stringhe di sei bit che cominciano con 1 e finiscono con zero (100000, 100010, 100100,....) e l'insieme di tutte le stringhe di lunghezza sette e che finiscono con 10.
H_a ed H_b sono detti modelli dello schema e sono elementi di spazi rispettivamente esa- ed epta-dimensionali.
Le differenti stringhe corrispondenti allo stesso modello si dicono istanze.
Lo schema è l'insieme del modello e del sottoinsieme di tutte le istanze che esso rappresenta.
Si definisce *ordine dello schema o(H)* il numero di bit definiti (2 per entrambi) e *lunghezza di definizione d(H)* la distanza tra i due bit definiti più esterni (5 e 1 rispettivamente).
Il numero di stringhe possibili è 2^l (l lunghezza stringa). Se consideriamo un certo sottoinsieme una stringa ha due possibilità, appartenervi o meno. Quindi i sottoinsiemi possibili saranno $2^{2^{\wedge l}}$ cioè 2^{2l}, mentre il numero di schemi possibili è 3^l.
Assumiamo l'ipotesi che gli operatori degli AG agiscano “vedendo” solo gli schemi[6] , cioè che siano proprio gli schemi i blocchi costitutivi ai quali si accennava avanti.
Chiaramente l'AG agisce trovandosi di fronte una stringa di lunghezza l. Questa è a sua volta istanza per più schemi, 2^l in particolare.
Ad esempio la stringa 11 è un'istanza dei quattro seguenti modelli : 1*,*1,**,11.
Così, una popolazione di n stringhe, contiene le istanze appartenenti ad un numero di schemi compreso tra 2^l e $n*2^l$, 2^l se le stringhe sono tutte uguali.

---

[6]. La definizione del tipo di schema è strettamente legata alla natura degli operatori che su questi stessi agiscono.

L'AG dunque procede nel calcolare esplicitamente il valore del fitness per ogni cromosoma.
Tuttavia, dato che la popolazione è generata casualmente, si può stimare che mediamente n/2 stringhe saranno istanze di

$$H_1 = 1{*}{*}{*}{*}\ldots{*}{*},$$

e le altre n/2 istanze di

$$H_0 = 0{*}{*}{*}{*}\ldots{*}{*}.$$

Una stima dell'idoneità della popolazione sarà quindi data dalla idoneità di uno dei due schemi H_1 o H_0 (cioè l'idoneità media di tutte le possibili istanze del schema).
Queste medie non sono calcolate, ma l'AG funziona in base all'aumento o alla diminuzione del numero di istanze di un determinato schema. Vogliamo capire qual'è la tipologia di schema che risulta più adatta, cioè lo schema al quale corrispondono più istanze (il numero degli schemi è ben più alto del numero di individui).
Valutiamo come varia il numero m(H,t) delle istanze x dello schema H all'istante t, al tempo t+1. Definiamo:

$$\hat{u}(H,t) = \frac{\sum_{x \in H} f(x)}{m(H,t)} \qquad (2.2)$$

l'idoneità media di tutte le istanze x di H al tempo t (somma delle idoneità di tutte le istanze dello schema fratto il loro numero totale, cioè m).
Se assumiamo che il numero di discendenti attesi per una certa stringa generica x è il rapporto tra l'idoneità della stringa f(x) e l'idoneità media della popolazione all'istante t $\bar{f}(t)$. Questo numero va sommato su tutte le stringhe x che sono presenti come istanze di H:

$$E(m(H,t+1)) = \frac{\sum_{x \in H} f(x)}{\bar{f}(t)} = \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t) \qquad (2.3)$$

quindi, con un solo passaggio otteniamo che il numero medio di istanze dipende dall'idoneità media delle istanze $\hat{u}(H,t)$che però ricordiamo, non viene calcolato. È un dato puramente statistico.

Ora ci chiediamo cosa è che fa cambiare il numero di istanze m dello schema. Abbiamo i due meccanismi di incrocio e di mutazione. Per ora assumiamo che questi siano distruttivi, e che possano al limite far diminuire il numero di istanze, e quindi cerchiamo qual'è la probabilità che uno schema H sopravviva ad un incrocio a punto singolo e ad una mutazione (cioè la probabilità che almeno uno dei discendenti sia ancora un'istanza dello stesso schema).
Affinché il crossing over distrugga lo schema, è necessario che avvenga all'interno della lunghezza di definizione. Quindi risultano più adatti gli schemi corti. Inoltre tanto più è lungo il cromosoma tanto minore è la probabilità che il crossing over distrugga lo schema.
Se l'incrocio avviene con probabilità p_c, la probabilità di sopravvivenza dello schema sarà al più:

$$S_c(H) \geq 1 - p_c(\frac{d(H)}{l-1}) \qquad (2.4)$$

con ovvio significato dei simboli.
Quello ottenuto è un limite inferiore perché può darsi che l'incrocio pur avvenendo lasci invariato lo schema (cioè un limite superiore della probabilità che lo schema sia distrutto) e risulta tanto più alto quanto più lo schema è corto.
Per quanto riguarda la mutazione, data p_m la probabilità che questa avvenga per un bit singolo, 1- p_m la probabilità che invece resti inalterato. Uno schema sopravvive se ogni bit che lo compone sopravvive:

$$S_m = (1 - p_m)^{o(H)} \qquad (2.5)$$

La probabilità di sopravvivere è maggiore per gli schemi di ordine minore.
Per la proliferazione di uno schema dunque, oltre ad avere un alto valore di percentuale di fitness, è indispensabile che contenga un piccolo numero di geni specifici non lontani l'uno dall'altro. Ciò, infatti, riduce la probabilità di distruggere lo schema durante la fase di riproduzione.
Utilizzando queste considerazioni modifichiamo l'equazione 2.3

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t)(1-p_c \frac{d(H)}{l-1})[(1-p_m)^{o(H)}] \qquad (2.6)$$

Cioè abbiam individuato un limite inferiore per la stima che volevamo trovare per i due motivi già citati, cioè il fatto di considerare incrocio e mutazione come processi esclusivamente distruttivi e il fatto che possono esserci degli incroci che rompono uno schema per riformarne uno eguale.
Gli schemi più adatti ad essere trasmessi sono quelli dunque più corti e con l'ordine minore, la cui idoneità media sia superiore a quella media generale.
Questi schemi crescono in maniera esponenziale, poiché ci si aspetta che questa popolazione, oltre a possedere un'idoneità sopra la media, viene incrementata al tempo t+1 di un valore circa uguale a $\frac{\hat{u}(H,t)}{\bar{f}(t)}$.

2.4 - IL RUOLO DEGLI OPERATORI SUI BLOCCHI

Una delle principali fonti della potenza di un AG è considerata proprio la capacità costruttiva del crossing over.
Infatti se gli schemi sono meglio trasmessi se corti e i basso ordine, il loro mescolamento porterà ad avere nuovi schemi altrettanto buoni o addirittura migliori e quindi, indirettamente geni altamente qualificati. Tale ipotesi è nota come *Ipotesi dei Blocchi Costruttivi*.
Tramite riproduzione incrociata dunque, l'algoritmo aumenta la probabilità che tali blocchi, provenendo da cromosomi diversi, si ritrovino in uno stesso cromosoma. Assumendo che l'associazione di

siffatti blocchi sia dunque vantaggiosa, dovrà anche ritenersi probabile la comparsa di un cromosoma (soluzione) eccellente per il problema in esame, in un tempo ragionevole.
Per quanto concerne la selezione dei cromosomi per la riproduzione invece, questa semplicemente indirizza gradualmente la procedura di campionamento verso le istanze degli schemi con idoneità superiore a quella media.
Il ruolo della mutazione sarebbe invece quello di assicurare una certa diversità all'interno della popolazione evitando, ad esempio, che tutte le stringe nella popolazione inizino col valore 1. La mutazione diventa una sorta di catalizzatore, se l'istanza ottenuta fa parte di uno schema più favorevole sarà conservata, altrimenti verrà eliminata.
Impedire un tale irrigidimento dell'algoritmo risulta di primaria importanza, poiché aumenta il volume dello spazio delle soluzioni sondato, sebbene da solo risulta insufficiente.

SVILUPPI ULTERIORI

La dimostrazione del teorema degli schemi è basata sull'ipotesi di codifica binaria, ma A.H.Wright nel 1991 l'ha estesa al caso di codifica con numeri reali; lo stesso Wright ha mostrato che una codifica reale è da preferirsi nel caso di problemi continui d'ottimizzazione. Successivamente Herrera e Lozano (1998) hanno presentato un'ampia rassegna di operatori genetici applicabili a cromosomi codificati mediante numeri reali, compresi vari tipi di operatori di crossing over.
Pertanto, il campo dei numeri reali costituisce ormai un'appropriata e consolidata forma di rappresentazione per gli algoritmi genetici in domini continui. Tuttavia, a causa di complessi fenomeni di interazione non lineare[7] tra gruppi di valori di una stringa rappresentante un individuo, non si può affermare con certezza che la combinazione di blocchi costitutivi altamente performanti sia sempre destinata a produrre individui ancora migliori. In altri termini, non sempre l'operazione genetica di crossing over produce risultati vantaggiosi, e anzi a volte può accadere che, a partire da una coppia di genitori molto promettenti, si ottenga un discendente non all'altezza delle aspettative.
La ricerca sugli AG è attualmente ricca di nuove proposte e sempre in aggiornamento.

[7] Tali comportamenti, noti come epistaticità, consistono nella presenza di geni il cui compito è quello di modificare gli altri geni secondo schemi gerarchici

3. LA SIMULAZIONE DELL'EVOLUZIONE

3.1 - I PROTOZOI DI PALMITER

Il concetto di selezione naturale si basa sull'ipotesi che, tra le diverse specie, quella che si afferma è quella che meglio riesce a sfruttare le risorse disponibili nell'ambiente.
È la natura dunque a selezionare la specie più adatta, quella con il corredo genetico più adatto all'ambiente nel quale vive.
Consideriamo un esempio abbastanza elementare, un programma molto

popolare in grado di simulare lo sviluppo di una colonia di protozoi o *Bugs* che cercano di sopravvivere all'interno di un ambientale bidimensionale nel quale il cibo (dei batteri immobili) viene distribuito a caso. Ogni batterio mangiato (per mangiare è sufficiente che il bug passi sopra il batterio) fornisce al bug una certa quantità di energia, tale energia viene spesa in seguito per muoversi.
I Bugs sono altresì in grado di riprodursi per fissione e di evolvere in maniera autonoma in base alla loro energia. Le uniche due condizioni imposte per ottenere che un Bug si riproduca sono che il Bug in questione sia adulto e che abbia un'energia sufficiente. Quando il Bug esaurisce la propria energia muore.
Il programma utilizzato permette all'utente di decidere quanta energia viene assorbita mangiando un batterio, qual'è l'età alla quale il Bug diviene adulto, qual'è il massimo valore della funzione di fitness (cioè il valore oltre il quale, anche mangiando ancora il bug non assorbe più energia) e a quale valore di tale funzione si ha la fissione del Bug.
In zone in cui c'è abbondanza di cibo un Bug può acquisire molta energia in pochi passaggi. D'altro canto può succedere che il Bug trovi scarsità di cibo per un lungo periodo, e quindi, che muoia.
Un Protozoo viene raffigurato come un puntino colorato, in grado di muoversi in sei direzioni, avanti (Forward), indietro (Backward), 60° destra (Right), 120° destra (Hard Rigth), 60° sinistra (Left), 120° sinistra (Hard Left).

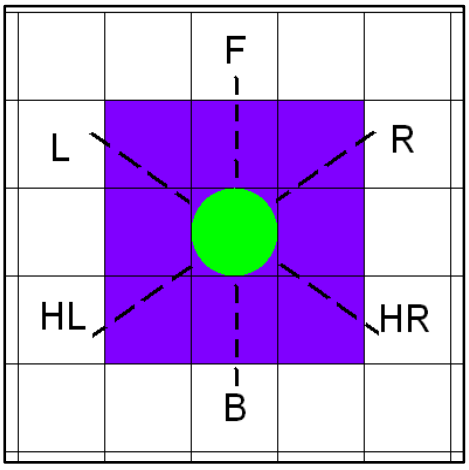

Fig 3.1 – Possibili cambiamenti di direzione per un Bug

Nell'implementazione dell'algoritmo queste lettere indicano piuttosto un cambiamento di direzione rispetto a quella attualmente mantenuta

dal Bug, corrispondente a F.
Il suo corredo genetico detta il percorso da seguire e consta appunto di una stringa di sei numeri interi, positivi o negativi.
Tali valori specificano la specie alla quale il Bug appartiene[8].

3.2 - OPERATORI DEI BUGS

FISSIONE

Il processo di riproduzione si ha per fissione: il Bug padre si scinde in due Bugs figli. Serve dunque un solo genitore.
L'operatore è inibito fintanto che il Bug padre non raggiunge età e livello di fitness prestabilito dall'utente per la riproduzione.
A quel punto l'operatore può agire, il programma cancella il Bug e al suo posto ne fa apparire altri due, ciascuno con metà dell'energia del genitore e un corredo genetico pressoché identico, in cui solo un gene risulta incrementato di un'unità in un figlio e diminuito nell'altro.

8. Nel caso di entità autoreplicanti all'interno di una struttura chimico-fisica soggetta a forti variazioni si usa la nozione di *quasispecie*, definita in termini di prossimità di certi valori nella stringa. Due Bugs appartengono alla stessa *quasispecie* se hanno il gene dominante all'interno dello stesso intervallo del cromosoma. Le mutazioni sono molto più comuni nei discendenti, rispetto a quelle delle specie.

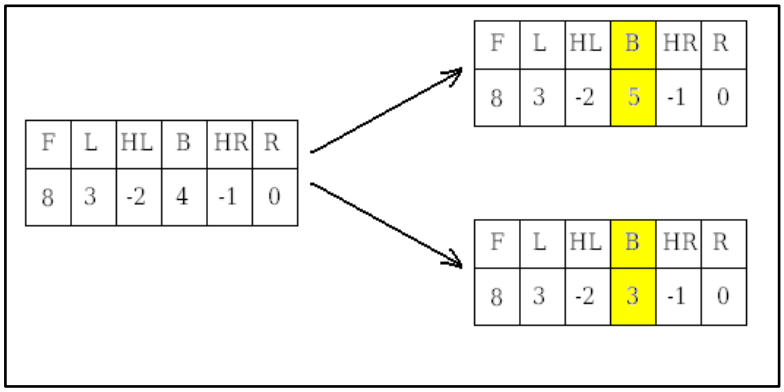

Fig 3.2 – Esempio di fissione di un Bug. Si noti come la variazione non comporti un grosso cambiamento nei figli, essendo avvenuta su un gene non dominante.

Questo programma è un esempio in cui l'algoritmo non sostituisce completamente la popolazione ad ogni iterazione. Agisce invece su ogni elemento in maniera indipendente dalla salute o dall'età degli altri.

OPERATORI DIREZIONALI

Gli operatori direzionali agiscono solo in base al corredo genetico che non varia nella vita di un Bug, indipendentemente quindi dal valore di fitness.
Ad ogni iterazione dell'algoritmo la posizione del Bug, definita sul reticolo dal vettore {i,j} viene aggiornata di un vettore $\overrightarrow{S_n}$nella posizione {i',j'} (nel sistema di riferimento del laboratorio).

$$\{i', j'\} = \{i, j\} + \overrightarrow{S_n} \qquad (3.1)$$

Il vettore$\overrightarrow{S_n}$,essendo definito sui vertici di un esagono approssimativamente regolare, assume valori diversi in base alla direzione. Possiamo dare la seguente parametrizzazione:

$$S_n \rightarrow \begin{Bmatrix} \{0,2\} & n=0 \\ \{2,1\} & n=1 \\ \{2,-1\} & n=2 \\ \{0,-2\} & n=3 \\ \{-2,-1\} & n=4 \\ \{-2,1\} & n=5 \end{Bmatrix} \quad (3.2)$$

L'indice n è modulare:

$$n = mod_6(k) \quad (3.4)$$

k è il valore sul quale, per definizione, agiscono gli operatori di direzione, incrementandolo.
$F(k) = k; L(k) = k+1; HL(k) = k+2; B(k) = k+3; HR(k) = k+4; R(k) = k+5$

Quando applichiamo uno dei 6 operatori direzionali il bug cambia la propria orientazione nel sistema di riferimento del laboratorio. Quindi l'azione di questi operatori si vede solo nel sistema del laboratorio mentre, nel sistema con gli assi centrati sulla posizione in cui ha agito l'ultima volta l'operatore direzionale, il Bug si muove sempre lungo la stessa direzione.
Ad esempio se il bug si trova nella posizione generica {i,j} e il programma sceglie l'operatore HR (n=4) il bug si sposta nella casella {i-2,j-1}.
Applicando l'operatore B l'indice n diventa 1 (mod_6 (4+3)= mod_6 (7)=1), quindi adesso il bug modifica la direzione di movimento rispetto al sistema di riferimento del laboratorio, spostandosi in {i-2,j-3}. Rispetto al sistema {x',y'} invece si è mosso lungo la direzione 5.
Le probabilità di movimento in una certa direzione vengono espresse normalizzando rispetto a tutte le altre possibilità.
Ad esempio la probabilità che un bug ha di muoversi in avanti è:

$$P_F = \frac{2^F}{2^F+2^B+2^R+2^{HR}+2^L+2^{HL}} \quad (3.3)$$

I valori vengono assegnati come esponenti in base due, onde evitare di avere valori negativi per le probabilità.
Un Bug con valore F=7 avrà un'alta probabilità di muoversi in avanti rispetto ad uno che ha piuttosto F=-4.
Il programma sceglie tra le 6 possibili direzioni in base a questi valori.

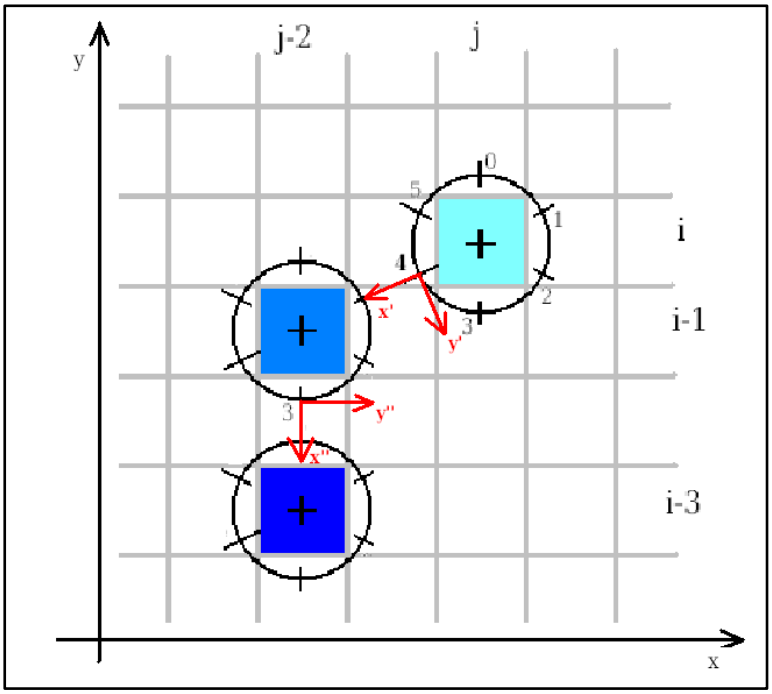

Fig 3.2 – Spostamento del Bug lungo il reticolo nel sistema di riferimento del laboratorio.

L'azione di S viene modificata da condizioni al contorno periodiche, per le quali se il quadrante è alto N, quando il bug raggiunge il limite superiore del quadrante, si ha:

$$\overrightarrow{S_0}(\{i,N\}) = \{i,2\} \tag{3.5}$$

e non

$$\overrightarrow{S_0}(\{i,N\}) = \{i,N+2\} \tag{3.5'}$$

Il discorso è identico sia per il limite inferiore che per le pareti laterali.

3.3 - SPECIE DIVERSE

Quando parliamo di *specie* di Bugs ci riferiamo ad agenti

caratterizzati dallo stesso comportamento dominante.
Il Bug non è capace di cambiare il proprio corredo genetico, al contrario la simulazione non sarebbe plausibile da un punto di vista biologico.
In questo modo si riesce a focalizzare l'attenzione sul problema fondamentale: la popolazione evolve attraverso la competizione, che è selettiva, e raggiunge la situazione che meglio si confa all'ambiente circostante, nello specifico la distribuzione del cibo.
All'inizializzazione il programma genera un certo numero di Bugs con una struttura genetica random.
Si avranno dunque molti Bugs che non hanno una direzione di moto preferenziale, cioè in cui nessuno dei geni ha un valore tale da predominare sugli altri e quindi il bug si muove in maniera *nervosa* (jitter).

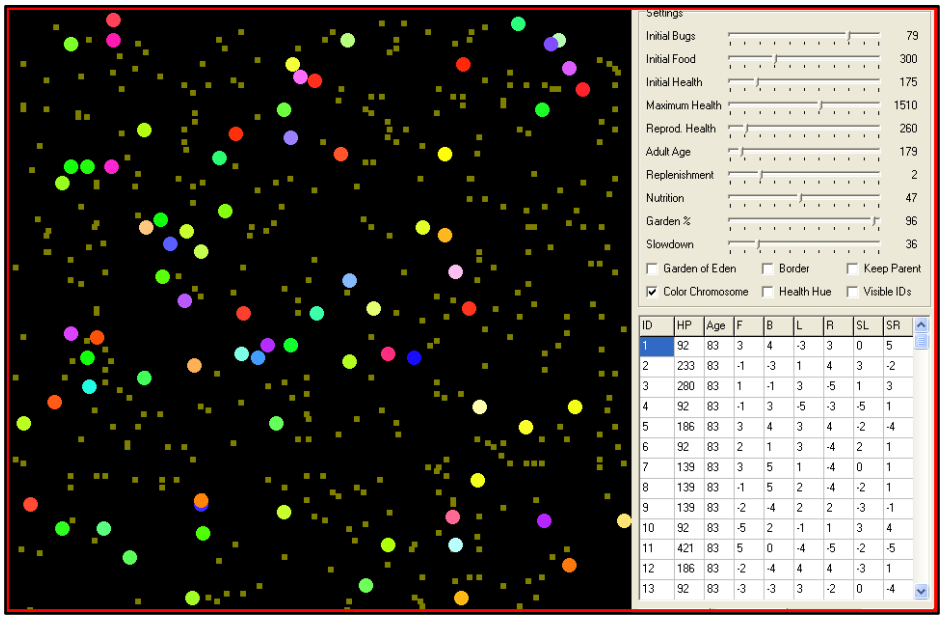

Fig 3.3 – Popolazione iniziale, generata casualmente.

Questi "jitterbugs" hanno un alto tasso di mortalità, dato che tendono a ritornare in posizioni occupate poco tempo prima, senza che il cibo abbia avuto la possibilità di rigenerarsi.
Alcuni però sopravvivono. Dopo un po' ci si rende conto che altri Bugs hanno cominciato a comportarsi differentemente, in particolare risulta che questi compiono un lungo tratto di percorso rettilineo prima di cambiare direzione, quindi sondano facilmente gran parte dello spazio

a disposizione in maniera quasi uniforme. Il gene dominante è il Forward oppure il Backward.
Proprio questa loro capacità gli permette di imporsi come popolazione dominante, cioè di prendere il sopravvento sulle altre. Tali Bugs hanno un colore blu.

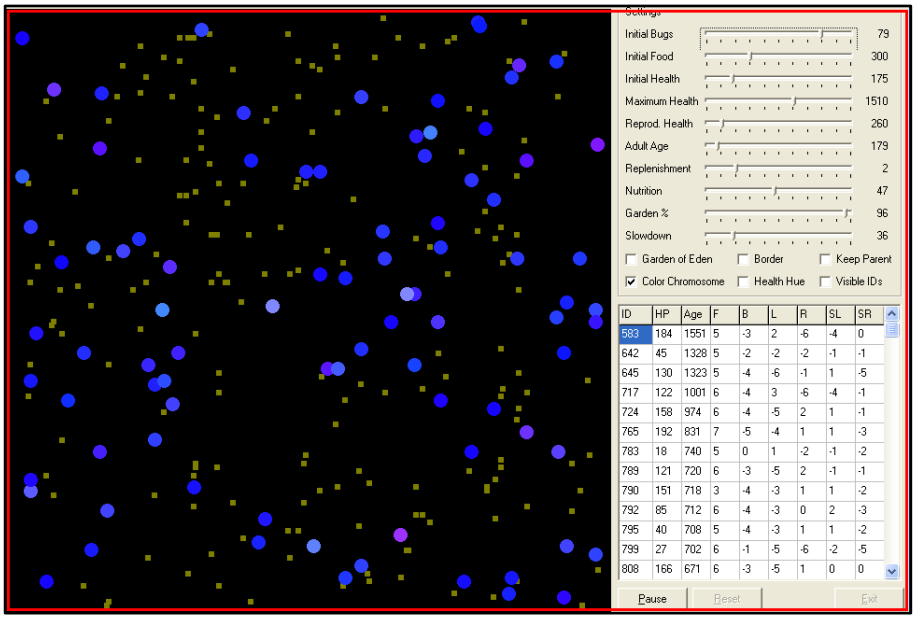

Fig 3.4 – Specie più adatta (distribuzione di cibo uniforme)

Tuttavia, sebbene la popolazione blu abbia preso il sopravvento, è possibile che, per via di una mutazione radicale (cioè mutazioni successive che casualmente rendono elevato il valore di un gene che non sia F), appaiano degli individui di altre specie. Potremmo parlare in questo caso di una vera e propria malattia genetica.
Una comune malattia è quella che affligge i *Twirliers* (rotatori), cioè dei Bugs con un forte spinta a muoversi in tondo, restando sempre all'interno della stessa zona con un movimento quasi circolare, periodico e rari spostamenti lineari.
In un ambiente con diffusione uniforme di cibo questi Bugs sono condannati a morire in breve tempo, ma il programma, come anticipato, ci offre una valida decodifica del concetto di come una mutazione genetica può essere il mezzo grazie al quale, in seguito ad una drastica modifica dell'ambiente, si ha l'affermarsi di una popolazione più adatta

al nuovo ambiente.

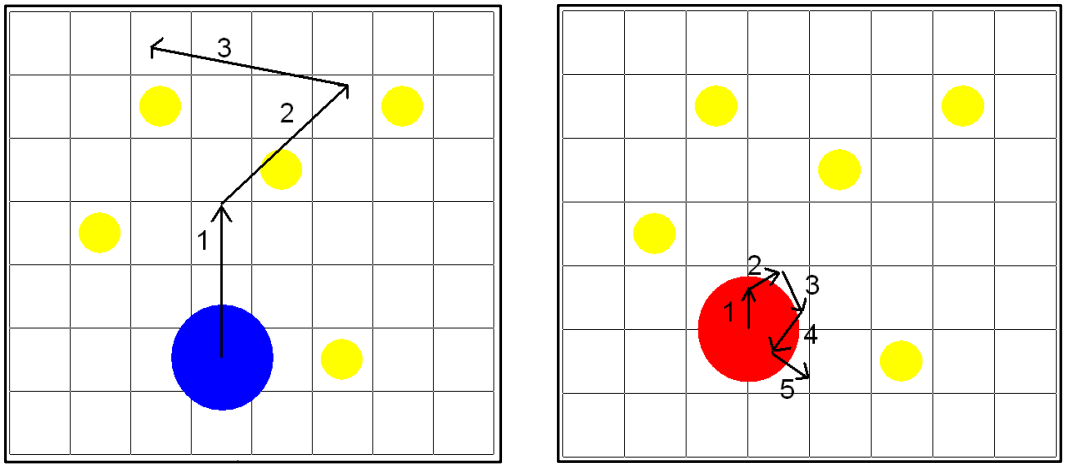

Fig 3.5 – Esempi per pattern di movimento.

Si può infatti sostituire al paesaggio normale (cibo diffuso uniformemente) un altro paesaggio con in più un riquadro centrale molto denso di cibo, detto "Giardino dell'Eden", di cui si può decidere la percentuale di cibo rispetto al totale.
In questo caso, la specie che si afferma sulle altre è quella capace di non allontanarsi troppo in senso longitudinale, cioè proprio i *Twirliers* a cui poc'anzi si accennava. Se i geni dominanti sono R o HR si ha colore rosso e moto diretto principalmente in senso orario, si i geni dominanti sono L o HL il colore è verde e il moto antiorario.
In ogni caso però, quando il cibo comincia a scarseggiare nel giardino dell'Eden, si ha una certa pressione ambientale che tende a far riaffiorare la popolazione blu. Questo dato dipende dalla disponibilità di cibo che l'utente decide di concentrare al centro.
Quello che normalmente è un difetto genetico risulta essere un vantaggio. Eliminando l'opzione giardino dell'Eden si ritorna verso il primo scenario.

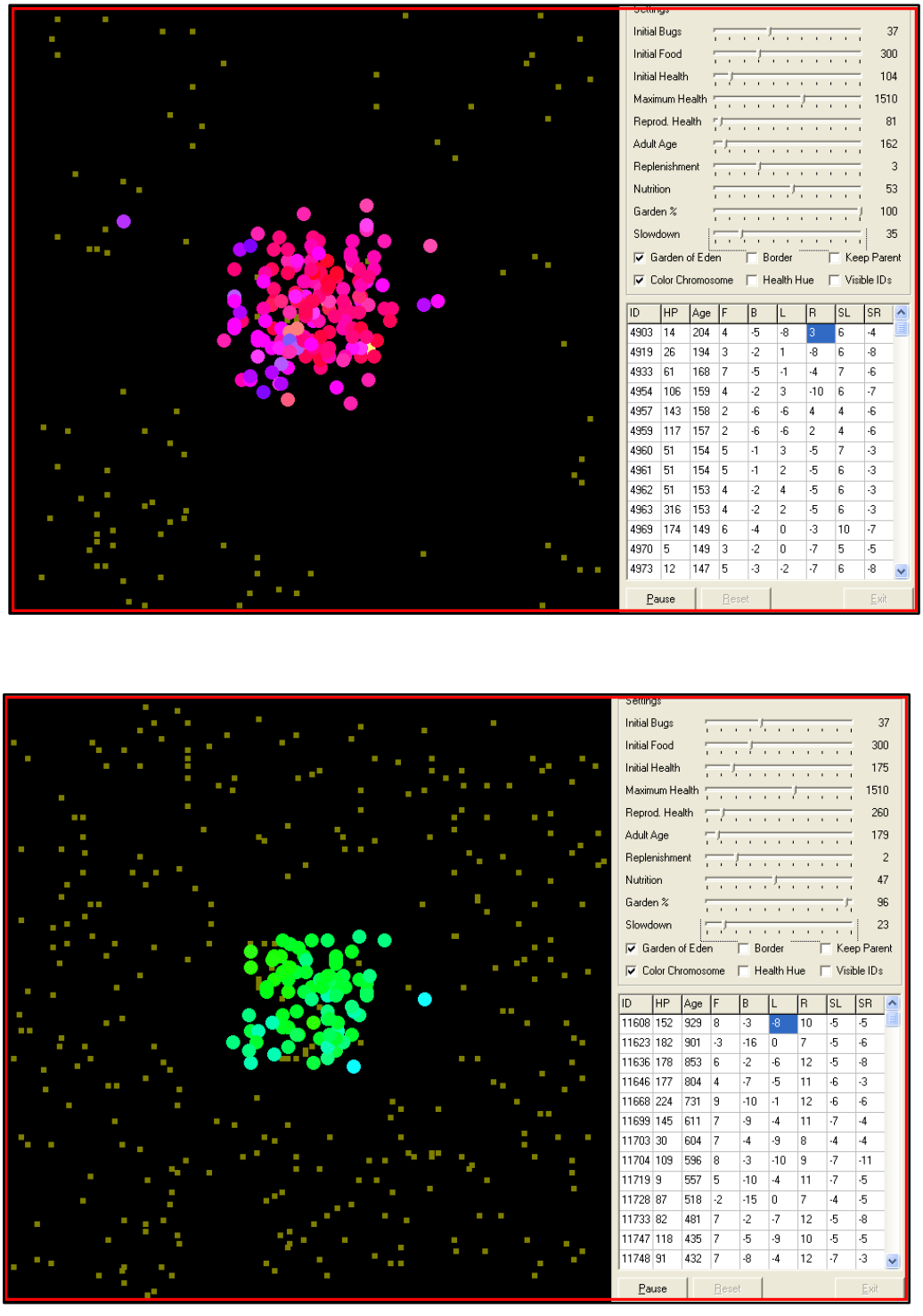

Fig 3.6 (A e B) – Specie più adatta (cibo nel Giardino dell'Eden)

Esistono tuttavia delle situazioni stazionarie di non equilibrio, nelle quali ponendo una percentuale di cibo nel giardino pari al 60% si ha una oscillazione regolare delle due popolazioni (rossa e blu o verde e blu) senza che l'operatore intervenga.
Questa situazione corrisponde ad una reazione chimica oscillante, o orologio chimico, strettamente legate alle oscillazioni di popolazione studiate da Lotka e Volterra.
Riconosciamo due situazioni, la prima in cui i Bugs rossi (o verdi) prevalgono e consumano velocemente il cibo all'interno del giardino. Quando il cibo all'interno del giardino risulta scarso la specie rossa comincia a decimarsi, mentre all'esterno i Bugs blu si affermano facilmente poiché il cibo ha avuto il tempo di rigenerarsi. A questo punto sono i Bugs blu a prevalere finché il giardino non ritorna florido a scapito del resto del paesaggio.
Può anche presentarsi il caso nel quale nel giardino dell'Eden vengono a coesistere le due specie *Twirliers*, quella verde con moto orario e quella rossa con moto antiorario.
Le due specie si contendono il cibo in maniera avida, ma essendo totalmente equivalenti, risultano entrambe candidate alla vittoria. Dopo un certo tempo non necessariamente breve, comunque si osserva che una specie riesce sempre a prendere il sopravvento sull'altra.
Tuttavia, partendo dalla stessa situazione (Bugs tutti identicamente blu) non si può dire con certezza verso quale situazione ci si dirige.
Possiamo affermare che, mentre con il cibo diffuso uniformemente esiste un'unica situazione di equilibrio stabile, che massimizza il valore del fitness, in questo caso le posizioni di equilibrio stabile sono due (intendendo per posizione il posto occupato in un asse delle specie).

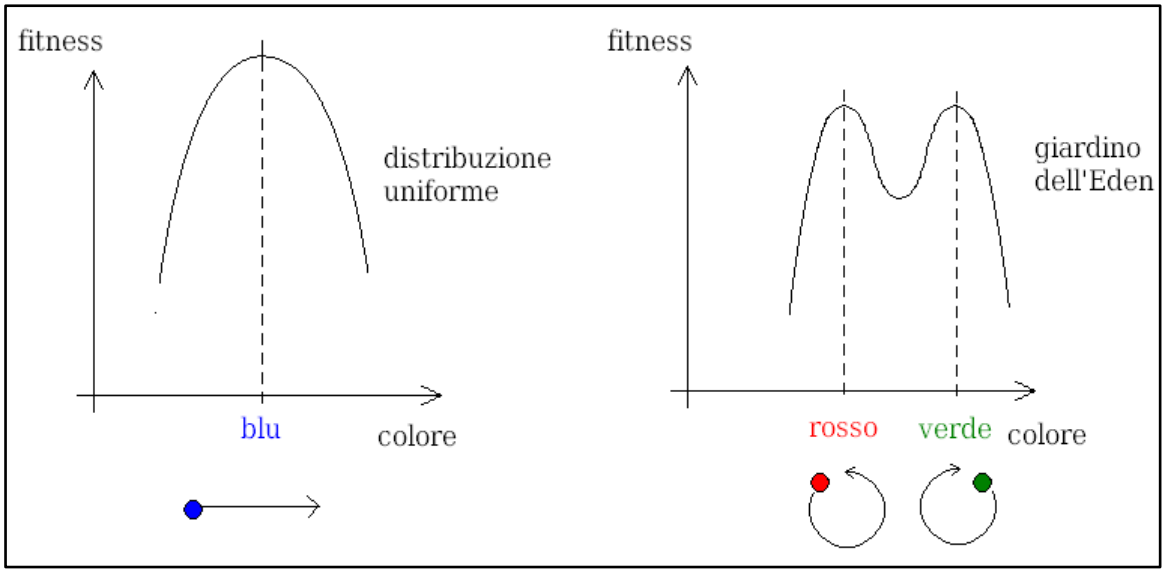

Fig 3.7 – Variazione del fitness

La ragione per la quale si ha un comportamento simile è da ricercarsi in una fluttuazione statistica che agisce a seguito di una rottura della simmetria.
Questa è un fenomeno deterministico che comporta uno sdoppiamento del picco nella funzione di fitness e genera la nascita di due massimi equivalenti.
La fluttuazione comporta quindi il sorpasso di una specie sull'altra.
Ribadiamo comunque che le due specie sono equivalenti, pertanto in questo caso la specie prevalente oltre ad essere la meglio adatta deve anche avere un certo grado di forutna.
Si può riuscire ad avere al più una distribuzione che consenta l'equilibrio tra le due, ma si tratterebbe di un equilibrio instabile.

La prima versione di questo programma è stata inventata da Michael Palmiter, alla fine degli anni '80 e salì velocemente alla ribalta. È un buon modello che sfrutta la selezione naturale per l'evoluzione mostrando effettivamente come sia l'ambiente a decidere qual'è la sequenza genetica che si afferma.
Questo programma è stato preso come paradigma per lo sviluppo successivo di altri software più complicati, in cui il corredo genetico è più complesso.

4. APPLICZIONI

Gli argomenti trattati finora potrebbero sembrare di stampo essenzialmente accademico. Si tenga tuttavia conto del fatto che in realtà esistono una moltitudine di problemi che possono essere risolti con questa metodologia.
Tale caratteristica deriva dalla generalità degli strumenti utilizzati, dalla semplicità delle regole usate, dallo sforzo atto ad ottenere il massimo grado di astrazione del problema e dalla capacità che poi i sistemi hanno di evolvere in base agli algoritmi imposti.
L'ambito tecnologico dove questi progressi sono più visibili è il campo dei videogiochi, in cui l'utente sempre più spesso ha a che fare con personaggi che evolvono autonomamente all'interno del loro mondo.
In altri casi l'evoluzione in ambienti virtuali viene usata per risolvere problemi reali ed estremamente attuali.

4.1 – ALCUNE APPLICAZIONI DEGLI AG

- Ottimizzazione numerica e combinatoria come la determinazione della miglior distribuzione degli elementi circuitali (cfr. 4.2) o la migliore distribuzione di carichi in una certa rete;

- Programmazione Automatica, cioè evoluzione di programmi atti a risolvere problemi specifici in Ecologia (coevoluzione del sistema parassita-ospite e sviluppo di una simbiosi cooperativa), in Economia (processi di innovazione, strategie di investimento a lungo termine), Scienze Sociali (evoluzione del comportamento nelle colonie di insetti),ecc..;

- Apprendimento Automatico, tra cui compiti di classificazione e predizione, come la possibilità di far evolvere alcuni parametri come i pesi per le reti neurali, le regole per i sistemi di classificazione o sensori per robot. Si pensi anche alla relazione che intercorre tra apprendimento individuale e evoluzione della specie.

L'elenco delle applicazioni è decisamente più vasto. Si dimostra come sia il software che l'hardware che possono evolvere e mutare in maniera autonoma in uno spazio virtuale.
Viene ampliato di seguito un esempio relativo ai problemi di ottimizzazione elettronica.

4.2 - UTILIZZO DELLA PROGRAMMAZIONE GENETICA IN ELETTRONICA

Nella maggior parte dei casi i problemi di elettronica hanno una natura combinatoriale, cioè per risolverli si deve sondare un enorme spazio di possibilità di al fine di trovare quella che meglio soddisfa determinate richieste.
Normalmente lo scienziato si avvicina al problema cercando di sintetizzare lo schema più adatto.
Un circuito elettronico viene sempre schematizzato in termini di pochi elementi fondamentali collegati tra loro: resistori, induttanze, capacità, diodi...
Consideriamo a titolo di esempio un problema concreto, la realizzazione di un filtro passa basso con frequenza di taglio f_T di 1k Hz.
Gli elementi primordiali sono banalmente dei circuiti in cui dei cavi conduttori connettono un input ad un output. A nostra disposizione ci sono resistori, induttanze e capacità che connettono via via più cavi tra loro (si noti come nessuno di questi abbia alcuna capacità di filtraggio).
Il primo passo è quello di “tradurre” l'organizzazione circuitale in un genotipo matematico, cioè una stringa che identifica il tipo di componente (R per resistenza, T per transistor...), il suo valore e i punti di connessione nella rete.

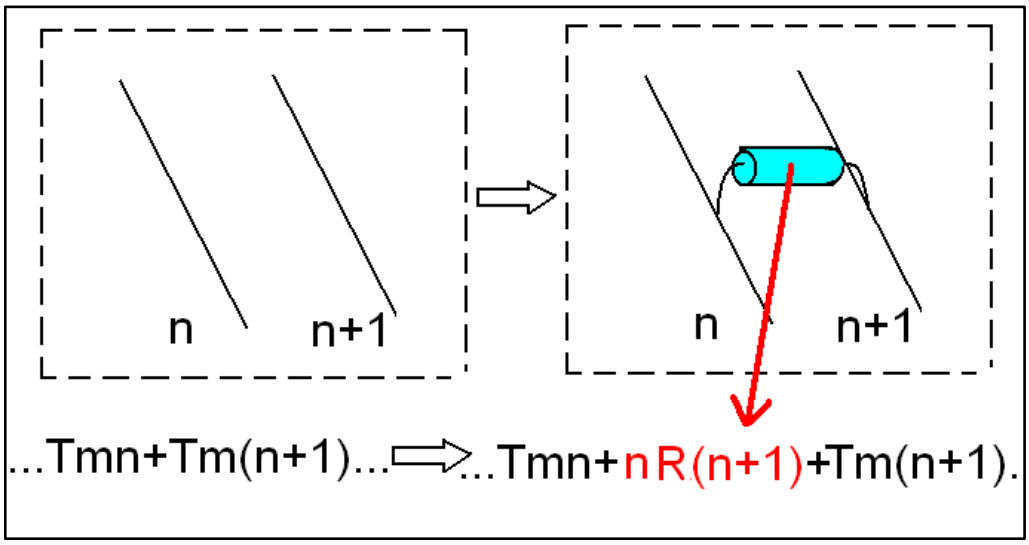

Fig 4.1 – Esempio di "traduzione"

L'algoritmo genetico consente ai componenti R,L, e C di creare collegamenti in serie o parallelo tra i cavi, connessioni a terra, cortocircuiti.
L'idoneità del circuito sarà valutata in base alla capacità del circuito di bloccare le frequenze superiori a f_T alla potenza massima.
Nella figura seguente si mostra un esempio di mescolamento tra due circuiti, affiancato dal mescolamento relativo alle due stringhe.
Sia G la lunghezza fissata del genoma, quindi il numero massimo di collegamenti possibili e sia P il numero di nodi del circuito.
Ogni nodo può essere collegato con ciascuno degli altri P-1 nodi, con un certo numero componenti minore di G (possibilità di collegamento in parallelo).
Ogni gene è composto da tre caratteri, due numerici (nodo di partenza e nodo di arrivo) e uno alfabetico centrale che indica il tipo di collegamento.
Assumiamo cinque possibili collegamenti con valori fissi[9]: resistenza R, capacità C, induttanza L, corto circuito CC e circuito aperto CA (più in generale sia n numero dei possibili componenti).

[9] Il valore numerico può anche cambiare, questo aumenta a dismisura il numero totale delle possibili soluzioni.

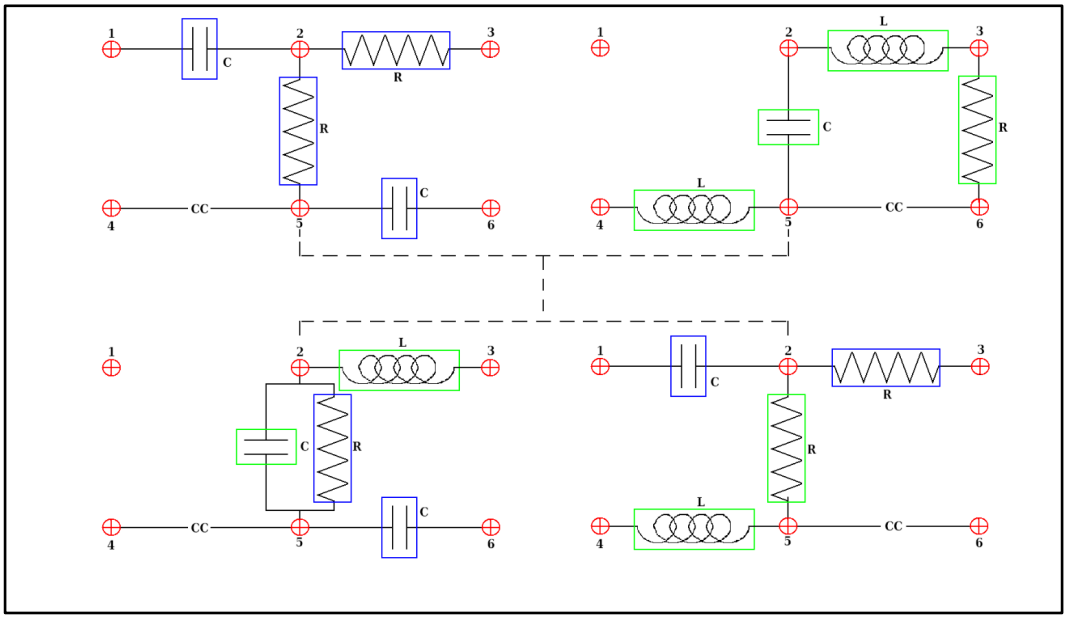

Fig 4.2 – Esempio di incrocio di circuiti (in alto) e di cromosomi equivalenti (in basso).

Ogni collegamento risulta indipendente dagli altri, quindi possiamo avere collegamenti in parallelo ma escludiamo la possibilità di avere collegamenti in serie, poiché questi comporterebbero la creazione di nuovi nodi, quindi implicitamente la variazione della lunghezza del genoma che noi abbiam fissato.
Cerchiamo dunque una stima del numero di possibili combinazioni:
ogni nodo di partenza può essere scelto in P modi, ogni nodo di arrivo in P-1 modi e il collegamento i 5 ulteriori modi differenti. Tutto questo va considerato per tutti i G elementi del genoma. Quindi il numero N di possibili combinazioni di G geni che l'algoritmo può simulare è:

$$N = (P * (P - 1) * n)^G \qquad (4.1)$$

Nel nostro caso se G=10 avremmo $N=150^{10}=1,5*10^{12}$ possibilità.
Osserviamo che il caso di collegamento banale CA non è riportato in figura.
Se, così come abbiamo fatto finora, trattiamo il circuito aperto come una delle cinque possibilità di collegamento, tra due nodi esiste sempre almeno un collegamento.
Il numero minimo di collegamenti possibili non ridondanti è

$$\frac{P*(P-1)}{2} \qquad (4.2)$$

Il cromosoma deve dunque essere lungo abbastanza da consentire questo numero, cioè un circuito composto solo da collegamenti semplici e nodi isolati.
Quindi

$$G \geq P\frac{(P-1)}{2} \qquad (4.3)$$

è condizione necessaria e il numero massimo di collegamenti in parallelo all'interno dello stesso circuito sarà:

$$G - \frac{P(P-1)}{2} \qquad (4.4)$$

Dopo un certo tempo si sarà ottenuto un gran numero di variazioni, e noi possiamo ritenere di aver raggiunto una soluzione soddisfacente.
Il processo fornisce sia la topologia del circuito sia le dimensioni del componente.
Del circuito soluzione noi non conosciamo i dettagli, in quanto ottenuto in base a combinazioni casuali di altri circuiti, ma sappiamo che il suo

funzionamento è soddisfacente rispetto alla condizione f_T=1k Hz.
D'altro canto risulta opportuno non imporre condizioni troppo stringenti al problema, al fine di lasciare la possibilità che emergano nuove possibilità.
L'algoritmo compone automaticamente i circuiti senza utilizzare alcun know-how del sistema in campo ingegneristico.

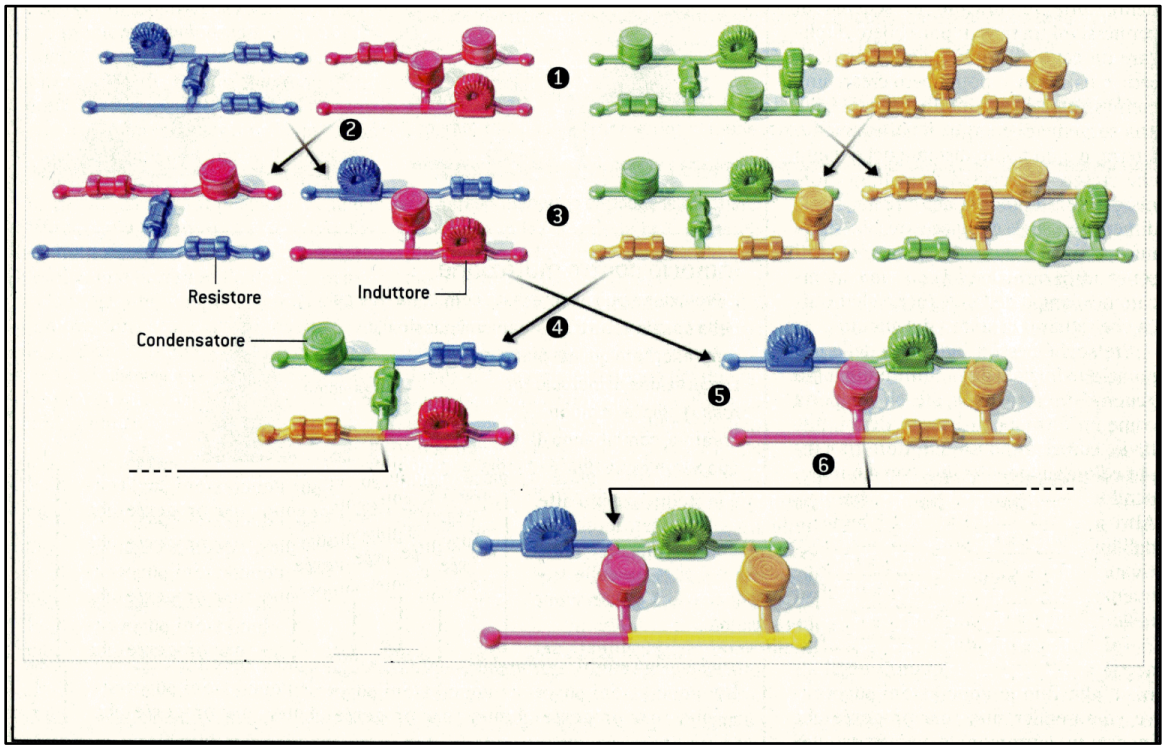

Fig 4.3 – Catena evolutiva
(tratto da J.H.Koza, M.A.Kane e M.J.Streeter; "L'evoluzione delle invenzioni", *Le Scienze*, Marzo 2003)

Lo stesso procedimento può essere esteso a strutture ben più complesse, come circuiti di filtraggio o di amplificazione. Nella figura seguente si mostra un altro esempio in cui un circuito ottenuto tramite AG vine confrontato qualitativamente con quello prodotto dall'uomo.
Non è assolutamente detto che la macchina abbia sempre la meglio sull'uomo. Senz'altro però le capacità di calcolo e di sintesi risultano incrementate di molto.

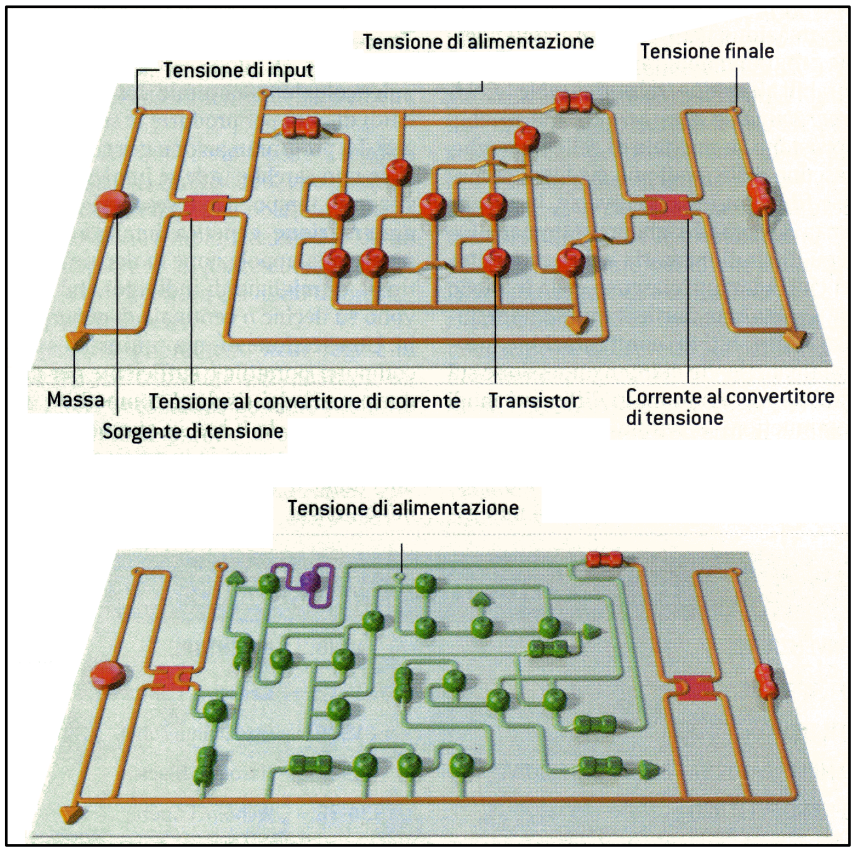

Fig 4.4 Schema creato dall'uomo (in alto) e schema evolutosi autonomamente (in basso)
(tratto da J.H.Koza, M.A.Kane e M.J.Streeter; "L'evoluzione delle invenzioni", *Le Scienze*, Marzo 2003)

Sfruttando tale approccio lo scienziato diventa l'utilizzatore di una macchina della quale si serve senza conoscerne la struttura.
Del sistema così ottenuto infatti non conosce né le d.d.p. tra i vari nodi, né la corrente che scorre nei vari rami.
Un approccio tradizionale su un circuito frutto dell'evoluzione può risultare un problema assi stimolante.
Svincolare gli strumenti elettronici dai problemi legati alla progettualità e permettergli contestualmente di evolvere liberamente porterà in breve tempo ad una nuova era dello sviluppo tecnologico.

Si tratta di un ulteriore esempio del comportamento dei sistemi complessi, nei quali il comportamento del gruppo non dipende più solamente dal comportamento dei singoli, ma anche dai vari tipi di interazione a più livelli.
Del resto non è la prima volta che metodi informatici riescono ad apportare simili contributi al miglioramento della tecnologia.
La propulsione viene dalle leggi naturali stesse, che diventano tanto più efficienti quanto più le risorse ambientali diventano limitate.

CONCLUSIONI

In questa tesi di laurea ho passato in rassegna diversi esempi dai quali emerge una nuova linea di pensiero adattata da alcuni in informatica ma anche in altri campi. Questa linea di pensiero consiste nell'emulare il migliore schema organizzativo a nostra disposizione: la Natura.
L'approccio computazionale è sempre più ispirato ai modelli biologici, dai quali c'è molto da imparare essendo il risultato migliore ottenuto dopo millenni di tentativi.
Inoltre le attuali capacità informatiche ci permettono di muoverci più abilmente, creando dei mondi virtuali in cui far crescere ed evolvere soluzioni a problemi altrimenti molto complessi da risolvere e simulando fenomeni difficili o addirittura impossibili da rappresentare sotto un qualsiasi sistema di equazioni.
Questo argomento è tra l'altro ancora essenzialmente inesplorato, esiste da poco più di quarant'anni e la maggior parte delle nozioni qui presentate sono ancora oggetto di studio.
Assieme ad altri settori di ricerca come le reti complesse, la genetica matematica e la fisica statistica applicata a questo approccio[10], potrebbe portare in prospettiva a pensare molti fenomeni, come l'evoluzione naturale o la diffusione delle informazioni, secondo nuovi modelli matematici.
Si consideri inoltre come fin qui si è analizzato solo l'incrocio tra due organismi aploidi e la mutazione. Molta ricerca si spinge nella direzione di incorporare nella sintesi degli AG anche meccanismi biologici più complicati quali la dominanza, la trasposizione, la differenziazione sessuale e la regolazione genetica. Mediante quest'ultima i geni degli organismi biologici si regolano reciprocamente in maniera indiretta. In questa maniera ad ogni situazione specifica solo determinati geni sono attivi, rendendo un sistema così complesso estremamente specifico e adattivo.
Inoltre sono oramai ben chiare le caratteristiche di apertura (possibilità di aumentare arbitrariamente la dimensione e la complessità del cromosoma), l'incapsulamento (la capacità di preservare dalla distruzione genetica parti utili e autonome del cromosoma), la regolazione gerarchica (la capacità di alcune parti del genotipo di regolarne altre) e una certa autonomia decisionale per la riproduzione dei cromosomi, come la scelta della posizione di applicazione dell'operatore di riproduzione.
Salendo ancora nella descrizione del mondo biologico, una

[10]Si intende a tal proposito lo studio di strutture macroscopiche che caratterizzano la popolazione nel complesso, senza tener conto del comportamento esatto dei singoli

modellizzazione del fenotipo potrebbe far si che i nostri AG si adattino ad un ambiente imprevedibile senza modificare la complessa codifica genotipica dell'individuo.

BIBLIOGRAFIA

- M.A. Bedau, TRENDS in Cognitive Sciences Vol.7 No.11 November 2003;
- M. Mitchell: *Introduzione agli algoritmi genetici*, Apogeo

scientifica (1998)

- A. K. Dewdney: *The Magic Machine: A Handbook of Computer Sorcery*, W.H. Freeman & Company (1990) ;
- J. R. Koza, M. A. Kane, M. J. Streeter, Le Scienze n°415 – Marzo 2003;
- J. Satinover: I*l cervello quantico*, Macro Edizioni (2002)
- S. Grand, D. Cliff, Autonomous Agents and Multi-Agent Systems, 1, 39–57 (1998)Kluwer Academic Publishers, Boston
- J.H. Holland: *Adaptation in Natural and Artificial Systems*. University of Michigan Press (1975), (seconda edizione: MIT press, 1992)
- A.H. Wright: *Genetic algorithms for real parameter optimization*,(1991) in C. Rawlins (a cura di), Foundations of Genetic Algorithms. Morgan Kafmann
- E. Schrödinger: *What is life? The physical aspect of the livin cell*, Cambridge University Press (1944), (ed. tradotta da Mario Ageno, 1995 Adelphi)