UNA BREVE INTRODUZIONE AGLI ALGORITMI GENETICI

S. Di Gregorio, D. D'Ambrosio, W.Spataro

1 Introduzione

A cavallo tra la fine degli anni '50 e l'inizio degli anni '60, molti ricercatori si interessarono dei sistemi naturali convinti che a essi ci si potesse ispirare per la realizzazione di nuovi algoritmi di ottimizzazione. Gli organismi viventi sono, infatti, ottimi risolutori di problemi (per esempio sono in grado di sopravvivere nel loro ambiente) e le loro abilità si sono il risultato dell'evoluzione naturale. L'evoluzione di una specie è regolata essenzialmente da due processi fondamentali: la selezione naturale e la riproduzione sessuale. La selezione naturale determina quali individui di una popolazione sopravvivono abbastanza per riprodursi (per esempio quelli in grado di riconoscere i predatori e di sfuggirgli), mentre la riproduzione sessuale determina la ricombinazione del materiale genetico dei genitori dando luogo a un'evoluzione molto più rapida di quella che si avrebbe se tutti i discendenti contenessero semplicemente una copia dei geni di un genitore, modificata di tanto in tanto per mutazione.

I primi tentativi volti a realizzare strumenti di ottimizzazione ispirati all'evoluzione naturale, come le Strategie Evolutive di Rechenberg [8] e la Programmazione Evolutiva di Fogel, Owens e Walsh [3], hanno dato scarsi risultati perchè i testi di biologia dei primi anni '60 ponevano l'accento sulla mutazione anzichè sul processo riproduttivo per la generazione di nuove combinazioni di geni. Un sviluppo importante si ebbe verso la metà degli anni '60 quando John H. Holland [5] propose gli Algoritmi Genetici (AG) che, per la prima volta, tenevano in considerazione in modo significativo il fenomeno della riproduzione sessuale.

In breve, un algoritmo genetico è un algoritmo iterativo che operna su una popolazione di individui. I membri della popolazione, che codificano soluzioni candidate di un dato problema, sono valutati tramite una funzione che determina quanto bene essi risolvano il problema e i migliori sono selezionati per la riproduzione e copiati nel così detto mating pool. Gli individui della nuova popolazione sono creati utilizzando semplici operatori random, ispirati alla riproduzione sessuale e alla mutazione. Il ciclo di valutazione, selezione, riproduzione sessuale e mutazione è iterato per un certo numero di generazioni, fino al raggiungimento di un dato criterio di fermata, come illustrato in figura 1.

In alcune applicazioni, gli AG trovano buone soluzioni in tempi ragionevoli. Comunque, in qualche caso, gli AG possono richiedere un gran numero di costose valu-

```
AG
{
      t=0;
      inizializza la popolazione P(0)
      valuta la popolazione P(0)
      mentre(non(criterio di fermata))
      {
            t=t+1
            crea il mating pool MP da P(t-1)
            crea la popolazione P(t) da MP
            valuta la popolazione P(t)
      }
}
```

Figura 1: Schema iterativo standard di un algoritmo genetico.

tazioni di funzioni e, a seconda del costo di ognuna, l'algoritmo genetico può impiegare giorni, mesi o anche anni per trovare una soluzione accettabile [2]. Tuttavia, gli AG sono algoritmi imbarazzantemente paralleli, nel senso che è molto semplice realizzarne implementazioni efficienti per calcolatori paralleli. Infatti gli AG lavorano con una popolazione di soluzioni indipendenti e questo rende molto semplice distribuire il carico computazionale su più processori [2] in modo che più valutazioni possano essere eseguite simultaneamente riducendo significativamente i tempi di calcolo.

2 Terminologia degli Algoritmi Genetici

Molti termini del linguaggio degli AG sono derivati dalla Biologia reale dove, tuttavia, identificano oggetti decisamente più complessi.

In Biologia, i cromosomi sono filamenti di DNA che fungono da progetto per l'organismo. Ogni cromosoma è composto da geni, ognuno dei quali codifica una particolare proteina; le proteine, a loro volta, determinano le caratteristiche specifiche dell'organismo, come ad esempio, il colore degli occhi. Le posizioni dei geni all'interno del cromosoma sone dette locus, mentre le diverse configurazioni delle caratteristiche specifiche dell'organismo codificate dai geni, per esempio blu o verde per il colore degli occhi, sono dette alleli. La maggior parte degli organismi presentano più di un cromosoma, il cui insieme è detto genoma: alcuni organismi, detti diploidi, hanno i cromosomi del genoma accoppiati, mentre altri, detti aploidi, presentano cromosomi spaiati. Il termine genotipo identifica l'insieme dei geni del genoma mentre il risultato finale dell'evoluzione fetale, cioè l'individuo, è detto fenotipo. Nella riproduzione sessuale si verifica il fenomeno della ricombinazione o dell'incrocio: il materiale genetico dei due genitori si ricombina generando un nuovo patrimonio completo per i discendenti. Talvolta, durante la ricombinazione, possono intervenire delle mutazioni casuali su singole parti del DNA; il fenomeno, abbastanza raro, prende il nome di mutazione.

Infine, l'idoneità (fitness) di un individuo rappresenta la probabilità che l'individuo viva abbastanza per riprodursi.

Negli AG, il termine cromosoma identifica la codifica di una soluzione candidata di una dato problema di ricerca. Holland propose per il suo modello una codifica binaria; in tal caso il cromosoma identifica una stringa di bit. I geni sono le parti costituenti del cromosoma e i valori che possono assumere sono detti alleli. Nel caso del'originario modello di Holland i geni sono i bit della stringa, mentre gli alleli possono essere 0 e 1. L'incrocio, o crossover, consiste solitamente nella ricombinazione del materiale genetico di due genitori aploidi composti da un solo cromosoma e la mutazione nella variazione casuale del valore degli alleli in ogni locus, o posizione, del cromosoma. Il termine fenotipo identifica, infine, il significato del cromosoma, ovvero la decodifica della soluzione candidata del problema dato. Poichè nella maggior parte delle applicazioni si utilizzano individui aploidi a singolo cromosoma, i termini genotipo, cromosoma e individuo sono considerati equivalenti. Inoltre, nei casi in cui la codifica del cromosoma rappresenti direttamente una soluzione candidata, per esempio in alcune applicazioni dove il cromosoma è una stringa di numeri reali anzichè di bit, anche i termini genotipo e fenotipo possono coincidere.

3 Il modello di Holland

L'originario modello proposto da Holland [5] opera su una popolazione di n stringhe di bit di lunghezza l fissata ($n, l \in \mathbb{N}$), di solito generate in modo casuale. Si osservi che l'insieme delle stringhe binarie di lunghezza l ha 2^l elementi e che tale numero, crescendo esponenzialmente al crescere di l, può essere anche molto grande. Tale insieme rappresenta lo spazio che l'algoritmo genetico deve esplorare per risolvere il problema di ricerca. Ogni stringa (genotipo) è la codifica binaria di una soluzione candidata (fenotipo) del problema che si vuole risolvere. Tramite una funzione f, detta di fitness, a ogni genotipo g_i (i=1,...,n) della popolazione iniziale $P(t = 0)$ è associato un valore $f_i = f(g_i)$, detto di fitness o di adattività, che rappresenta la capacità dell'individuo di risolvere il problema dato. Per determinare il valore di adattività, infatti, la funzione di fitness riceve in input un genotipo, lo decodifica nel corrispondente fenotipo e lo testa sul problema dato.

Terminata la valutazione degli individui della popolazione iniziale, viene generata una nuova popolazione $P(t+1)$ di n nuove soluzioni candidate applicando gli operatori di selezione, crossover, mutazione e inversione.

3.1 Selezione

Proporzionalmente al valore di fitness f_i, ai genotipi g_i sono associate le probabilità

$$p_{selection,i} = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1}$$

utilizzate per costruire una sorta di roulette di probabilità. Per esempio, se la popolazione è composta dagli $n = 4$ individui A_1, A_2, A_3 e A_4, con le rispettive probabilità

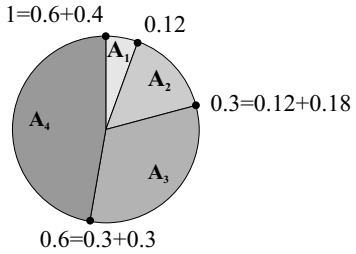

Figura 2: Esempio di roulette per l'operatore di selezione.

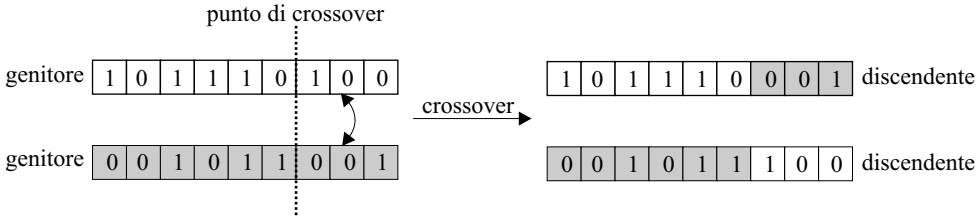

Figura 3: Esempio di crossover.

di selezione $p_{selection,1} = 0.12$, $p_{selection,2} = 0.18$, $p_{selection,3} = 0.3$ e $p_{selection,4} = 0.4$, la corrispondente roulette avrà la forma mostrata in figura 2. L'operatore di selezione genera un numero casuale $c \in [0, 1]$ e seleziona l'individuo la cui fetta di roulette ne contiene il valore. Per esempio se $c = 0.78$, con riferimento alla figura 2, l'individuo selezionato è A_4 poichè la fetta di roulette relativa ad A_4 specificata i valori dell'intervallo $[0.6, 1]$ e il valore di c ricade in tale intervallo. Quando un individuo è selezionato ne viene creata una copia e quest'ultima viene inserita nel così detto, mating pool. Una volta che il mating pool è riempito con n copie di individui della popolazione $P(t = 0)$, gli individui della nuova popolazione $P(t + 1)$ sono ottenuti come loro discendenti attraverso gli operatori di crossover, mutazione e inversione. L'operatore di selezione determina, dunque, quali individui della vecchia popolazione hanno la possibilità di generare dei discendenti e poichè gli individui con fitness alta sono quelli favoriti, potendo contare un numero di copie maggiore rispetto a quelli con fitness più bassa, l'operatore di selezione gioca, nel contesto dell'algoritmo genetico, il ruolo della selezione naturale per gli organismi viventi.

3.2 Crossover

Si scelgono a caso due individui nel mating pool, detti genitori, e un punto di taglio, detto punto di crossover, su di essi; le porzioni di genotipo alla destra del punto di crossover vengono scambiate generando due discendenti, come mostrato in figura 3. L'operatore di crossover è applicato, in accordo a una prefissata probabilità $p_{crossover}$, $\frac{n}{2}$ volte in modo da ottenere n discendenti; nel caso in cui il crossover non sia applicato, i discendenti coincidono con i genitori. Si osservi che, così come l'operatore di selezione

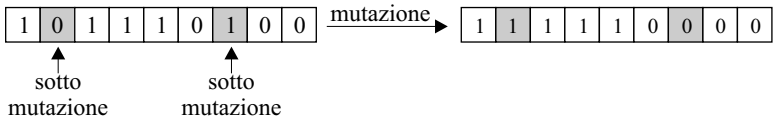

Figura 4: Esempio di mutazione.

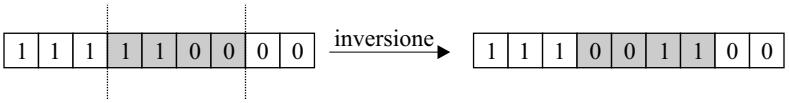

Figura 5: Esempio d'inversione.

gioca, nel contesto dell'algoritmo genetico, il ruolo della selezione naturale, il crossover rappresenta una metafora della riproduzione sessuale in cui il materiale genetico dei discendenti risulta essere una ricombinazione di quello dei genitori.

3.3 Mutazione

In funzione di una prefissata e usualmente piccola probabilità $p_{mutation}$, il valore dei bit di ogni individuo viene cambiato (da 0 in 1 o viceversa), come illustrato in figura 4. L'operatore di mutazione rappresenta il fenomeno genetico della rara variazione di elementi del genoma degli esseri viventi durante l'evoluzione.

3.4 Inversione

L'operatore d'inversione, ispirato come i precedenti a un analogo meccanismo biologico, in accordo a una prefissata probabilità $p_{inversion}$, sceglie casualmente due punti nella stringa che codifica l'individuo e inverte i bit tra le due posizioni, come illustrato in figura 5. In biologia il significato di un gene spesso non dipende dalla sua posizione nel cromosoma e, di conseguenza, l'inversione ne lascia invariata la semantica. Purtroppo questo non è vero per la maggior parte degli algoritmi genetici e l'uso di tale operatore porta inevitabilmente a complicazioni a volte anche molto significative. Per tale ragione l'inversione è raramente utilizzata sia nelle applicazioni pratiche che negli studi teorici.

4 Fondamenti teorici degli Algoritmi Genetici

Negli anni '70 John Holland [5] ottenne alcuni importanti risultati teorici: introdusse il concetto di schema (o modello di similarità) e di parallelismo implicito e dimostrò il teorema oggi noto come teorema fondamentale degli Algoritmi Genetici.

4.1 Schemi e parallelismo implicito

Definizione 4.1. Sia $V = \{0, 1\}$ l'alfabeto attraverso cui sono costruiti gli individui dell'algoritmo genetico. Si definisce schema una stringa della stessa lunghezza l degli individui dell'algoritmo genetico, costruiti sull'alfabeto esteso $V+ = \{0, 1, *\}$.

Osservazione 4.1. Il numero di stringhe che si possono costruire su V è 2^l, mentre il numero di schemi è $(2+1)^l$. Per esempio se $l = 5$, si avranno $k^l = 2^5 = 32$ individui e $(k+1)^l = 3^5 = 243$ schemi.

Uno schema rappresenta un insieme di stringhe che hanno un qualsiasi simbolo di V nelle corrispondenti posizioni in cui compare il simbolo $*$ o, equivalentemente, rappresenta un insieme di stringhe con caratteristiche comuni nelle corrispondenti posizioni in cui non compare il simbolo $*$.

Esempio 4.1. Lo schema $H_1 = *0000$ rappresenta l'insieme costituito dalle due stringhe aventi uno dei due simboli di V nella posizione 0 e il sibolo 0 nelle posizioni 2, 3, 4 e 5, cioè

$$H_1 = \{00000, 10000\}$$

Lo schema $H_2 = *111*$ rappresenta l'insieme costituito dalle quattro stringhe aventi uno dei due simboli di V nelle posizioni 1 e 5 e il simbolo 1 nelle posizioni 2, 3 e 4, cioè

$$H_2 = \{01110, 01111, 11110, 11111\}$$

Lo schema $H_3 = 0 * 1 * *$ rappresenta l'insieme costituito dalle otto stringhe aventi uno dei due simboli di V nelle posizioni 2, 4 e 3, uno 0 nella posizione 1 e un 1 nella posizione 3, cioè

$$H_3 = \{00100, 00101, 00110, 00111, 01100, 01101, 01110, 01111\}$$

.

Per comprendere meglio il significato di schema consideriamo il seguente esempio.

Esempio 4.2. Sia $V = \mathbb{R} + \cup\{0\}$, ed $l = 3$. Allora lo spazio di ricerca dell'algoritmo genetico è $S = \mathbb{R}^3 + \cup\{\mathbf{0}\}$.
Lo schema $H_1 = **0$ rappresenta il sottoinsieme di S formato dai punti del piano $z = 0$ (figura 6). Infatti i simboli $*$ nella prima e seconda posizione dello schema indicano che le coordinate x e y possono assumere un qualsiasi valore di $V = \mathbb{R} + \cup\{0\}$, mentre la coordinata z è fissata a 0.
Lo schema $H_2 = **1$ rappresenta il sottoinsieme di S formato dai punti del piano $z = 1$ (figura 7). Infatti, come prima, i simboli $*$ nella prima e seconda posizione dello schema indicano che le coordinate x e y possono assumere un qualsiasi valore di $V = \mathbb{R} + \cup\{0\}$, mentre la coordinata z è fissata a 1.
Lo schema $H_3 = 1 * 1$ rappresenta il sottoinsieme di S formato dai punti della retta $(x = 1) \cap (z = 1)$ (figura 8). Infatti il simbolo $*$ nella posizione 2 dello schema indica che la coordinata y può assumere un qualsiasi valore di $V = \mathbb{R} + \cup\{0\}$, mentre le coordinate x e z sono fissate a 1.

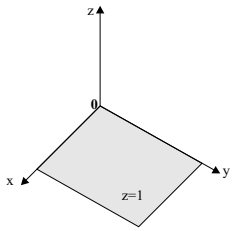

Figura 6: Il sottoinsieme $z = 0$ dello spazio di ricerca S dell'algoritmo genetico definito dallo schema $H_1 = **0$.

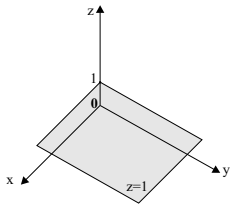

Figura 7: Il sottoinsieme $z = 1$ dello spazio di ricerca S dell'algoritmo genetico definito dallo schema $H_2 = **1$.

Gli schemi rappresentano, dunque, sottoinsiemi dello spazio di ricerca dell'algoritmo genetico.

Teorema 4.1. *Il numero di schemi di un algoritmo genetico presenti in una popolazione di n individui di lunghezza l costruiti sull'alfabeto $V = \{0, 1\}$ è compreso tra 2^l e $n2^l$.*

Dimostrazione. Consideriamo una generica stringa $A = a_1 a_2 \dots a_l$ tale che $a_i \in V = \{0, 1\} \quad \forall i \in \{1, 2, \dots, l\}$. Essa è un membro di ogni schema che abbia nelle corrispondenti posizioni di A o lo stesso simbolo di A oppure il simbolo $*$. Per esempio A è membro dello schema $H = *a_2 * * \dots * a_l$. Dunque, ogni schema che contenga A è vincolato ad avere in ogni posizione uno tra due simboli: il corrispondente simbolo di A, oppure $*$. Il loro numero è, pertanto, 2^l. Può succedere che, per esempio quando tutte le stringhe della popolazione sono ugluali, tutti gli individui siano membri degli stessi schemi; in tal caso il numero di schemi della popolazione è 2^l. All'opposto, quando tutte le stringhe della popolazione sono membri di schemi diversi, il loro numero è $n2^l$. □

Il teorema precedente afferma che l'algoritmo genetico, pur elaborando esplicitamente solo n individui, processa implicitamente un numero molto maggiore di schemi. Questo significa che è garantita la presenza di qualche soluzione candidata in un numero compreso tra 2^l e $n2^l$ sottoinsiemi differenti dello spazio di ricerca, garantendo un'esplorazione abbastanza efficace dello spazio delle soluzioni del problema. Tale caratteristica è chiamata *parallelismo implicito*.

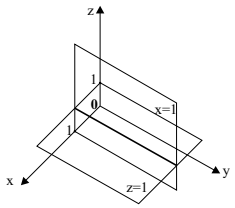

Figura 8: Il sottoinsieme $(x = 1) \cap (z = 1)$ dello spazio di ricerca S dell'algoritmo genetico definito dallo schema $H_3 = 1 * 1$.

4.2 Il teorema fondamentale degli Algoritmi Genetici

Definizione 4.2. Si definisce ordine di uno schema H, e si indica con $o(H)$, il numero di posizioni fisse presenti nel modello.

Esempio 4.3.

$$o(011 * 1 * *) = 4$$

$$o(0 * * * * * *) = 1$$

Definizione 4.3. Si definisce lunghezza caratteristica di uno schema H, e si indica con $\delta(H)$, la distanza tra la prima e l'ultima posizione fissa dello schema.

Esempio 4.4.

$$\delta(h_1h_2h_3h_4h_5h_6h_7) = \delta(011 * 1 * *) = 5 - 1 = 4$$

$$\delta(h_1h_2h_3h_4h_5h_6h_7) = \delta(0 * * * * * *) = 1 - 1 = 0$$

Consideriamo ora gli effetti della selezione, del crossover e della mutazione sul numero di rappresentanti degli schemi dell'algoritmo genetico. Supponiamo che a una data generazione t esistano $m = m(H, t)$ rappresentanti di un particolare schema H nella popolazione P(t). Come descritto nel paragrafo 3, formula 1, ogni individuo, quindi anche ogni rappresentante di H, è selezionato per la riproduzione con probabilità $p_{selection,i} = \frac{f_i}{\sum_{j=1}^{n} f_j}$. Dunque, la probabilità che una stringa dello schema venga selezionata a ogni tentativo è

$$\sum_{i=1}^{m} p_{selection,i} = \frac{\sum_{i=1}^{m} f_i}{\sum_{j=1}^{n} f_j} \tag{2}$$

Poichè l'operatore di selezione è eseguito n volte, il numero atteso di rappresentanti di H è

$$m(H, t+1) = n \sum_{i=1}^{m} p_{selection,i} = n \frac{\sum_{i=1}^{m} f_i}{\sum_{j=1}^{n} f_j} = \frac{\sum_{i=1}^{m} f_i}{\bar{f}} \tag{3}$$

dove $\bar{f}$ è la fitness media della popolazione $P(t)$. Moltiplicando ambo i membri per $m(H,t)$ si ha

$$m(H,t+1) = \frac{m(H,t)}{m(H,t)} \frac{\sum_{i=1}^{m} f_i}{\bar{f}} = m(H,t) \frac{f(H)}{\bar{f}} \tag{4}$$

dove $f(H)$ è la fitness media degli individui rappresentanti dello schema H. La formula 4 indica che se la fitness media dei rappresentanti di uno schema H è maggiore della fitness media della popolazione, allora il numero di rappresentanti dello schema aumenta, altrimenti diminuisce. Per determinare la velocità con cui varia il numero di rappresentanti scriviamo

$$f(H) = \bar{f} + c\bar{f} \tag{5}$$

dove c è un costante reale, minore di uno se la fitness media dello schema è minore della fitness media della popolazione, maggiore di uno altrimenti. Allora

$$m(H,t+1) = m(H,t) \frac{\bar{f} + c\bar{f}}{\bar{f}} = m(H,t)(1+c) \tag{6}$$

per cui si avrà

$$m(H,t) = m(H,t-1)(1+c) = m(H,t-2)(1+c)^2 = \ldots$$

$$\ldots = m(H,0)(1+c)^t \tag{7}$$

Dunque, il numero di rappresentanti di uno schema cresce, o decresce, esponenzialmente durante le generazioni dell'algoritmo genetico a seconda che la fitness media dello schema sia maggiore o minore della fitness media della popolazione

Consideriamo ora un individio A e due schemi, H_1 e H_2, di cui A sia un rappresentante

$$A = 0111000$$

$$H_1 = *1****0$$

$$H_2 = ***10**$$

e supponiamo che A sia sogetto a crossover con punto di taglio nella posizione 3, cioè dopo il terzo gene

$$A = 011 \mid 1000$$

$$H_1 = *1* \mid ***0$$

$$H_2 = *** \mid 10**$$

Osserviamo che i discendenti di A non saranno rappresentanti dello schema H_1, a meno che A non si incroci con un individuo che abbia anch'esso un 1 nella posizione 2 e uno zero nella posizione 7. Al contrario, almeno un discendente di A sarà ancora rappresentante di H_2. Lo schema H_1 ha minori probabilità di sopravvivere al crossover rispetto ad H_2 perchè ha maggiori probabilità che il punto di taglio cada tra le due posizioni fisse estreme. Infatti $\delta(H_1) = 7 - 2 = 5$, e il punto di taglio può cadere con

eguale probabilità in uno degli $l - 1 = 7 - 1 = 6$ posizioni. Allora le probabilità che H_1 e H_2 vengano distrutti sono

$$p_d(H_1) \leq \frac{\delta(H_1)}{l-1} = \frac{5}{6}$$

$$p_d(H_2) \leq \frac{\delta(H_2)}{l-1} = \frac{1}{6}$$

Pertanto, definiamo la probabilità che uno scema H venga distrutto come

$$p_d(H) \leq \frac{\delta(H)}{l-1} \tag{8}$$

Se il crossover è eseguito con probabilità $p_{crossover}$, la formula precedente diventa

$$p_d(H) \leq p_{crossover}\frac{\delta(H)}{l-1} \tag{9}$$

La probabilità di sopravvivenza di uno schema H sarà, dunque

$$P_s \geq 1 - p_d = 1 - p_{crossover}\frac{\delta(H)}{l-1} \tag{10}$$

L'espressione precedente afferma che gli schemi che anno alte probabilità di sopravvivere al crossover sono quelle che hanno lunghezze caratteristiche piccole. Moltiplicando la 4 con la 10 otteniamo il numero atteso di rappresentanti dello schema H per effetto della selezione e del crossover:

$$m(H, t+1) >= m(H, t)\frac{f(H)}{\bar{f}}(1 - p_{crossover}\frac{\delta(H)}{l-1}) \tag{11}$$

Affinchè uno schema H sopravviva anche all'operatore di mutazione, tutte le posizioni fisse $o(H)$ devono rimanere immutate. La probabilità di sopravvivenza sarà, dunque:

$$p'_s = (1 - p_{mutation})^{o(H)} \tag{12}$$

se $p_{mutation} \ll 1$, allora

$$p'_s = (1 - p_{mutation})^{o(H)} \approx 1 - o(H)p_{mutation} \tag{13}$$

Moltiplicando la 11 per la 13 otteniamo il numero atteso di rappresentanti dello schema H per effetto della selezione del crossover e della mutazione:

$$m(H, t+1) >= m(H, t)\frac{f(H)}{\bar{f}}(1 - p_{crossover}\frac{\delta(H)}{l-1})(1 - o(H)p_{mutation}) =$$

$$= m(H, t)\frac{f(H)}{\bar{f}}(1 - o(H)p_{mutation} - p_{crossover}\frac{\delta(H)}{l-1} +$$

$$+p_{crossover}\frac{\delta(H)}{l-1}o(H)p_{mutation}) \tag{14}$$

Il termine $p_{crossover}\frac{\delta(H)}{l-1}o(H)p_{mutation}$ si può trascurare perchè abbastanza piccolo, per cui la formula precedente diventa

$$m(H,t+1) >= m(H,t)\frac{f(H)}{\bar{f}}(1-p_{crossover}\frac{\delta(H)}{l-1}-o(H)p_{mutation}) \tag{15}$$

Il risultato precedente prende il nome di *teorema fondamentale degli algoritmi genetici*. Il numero atteso di rappresentanti di schemi caratterizzati da fitness media alta e lunghezza caratteristica piccola, chiamati da Goldberg Blocchi Costituenti [4], aumenta esponenzialmente (6) di generazione in generazione.

In conclusione, l'algoritmo genetico ha la capacità di esplorare un numero compreso tra 2^l e $n2^l$ schemi per ogni generazione (parallelismo implicito) e di concentrare il maggior numero di individui nelle zone dello spazio di ricerca più promettenti poichè premia gli schemi caratterizzati da fitness alta (teorema fondamentale degli algoritmi genetici).

5 Varianti del modello di Holland

In linea di principio è sempre possibile codificare le soluzioni candidate di un problema di ricerca attraverso stringhe binarie. Tuttavia, per la risoluzione di alcuni problemi risulta più naturale utilizzare rappresentazioni di livello più alto e definire operatori di crossover e mutazione in grado di operare su tali rappresentazioni. Nelle applicazioni pratiche le codifiche più diffuse sono quella binaria, quella basata su numeri reali e la codifica ad albero.

Scelta la codifica e adattati gli operatori di crossover e mutazione, occorre definire un'operatore di selezione opportuno. Infatti, una selezione troppo forte puo dar luogo alla predominanza di qualche individuo con fitness particolarmente alta e condurre l'algoritmo genetico in un ottimo locale da cui difficilmente riuscirà a uscire. D'Altro canto, una pressione selettiva troppo debole può dar luogo a un significativo aumento del tempo d'esecuzione necessario a trovare una soluzione accettabile.

5.1 Codifica binaria

La codifica binaria, sia per motivi storici, sia perchè i risultati teorici più importanti sono stati derivati attraverso modelli basati su tale codifica, è probabilmente la più utilizzata nelle applicazioni degli AG.

La struttura dati utilizzata è un vettore di bit di lunghezza l, cui corrisponde uno spazio di ricerca di 2^l possibili soluzioni. L'uso della codifica binaria richiede la specificazione di una funzione che decodifichi il genotipo, o parti di esso, in uno o più valori reali. L'equazione seguente decodifica un genotipo binario g di lunghezza l nel

Valore intero	Codifica binaria	Codifica grigia
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabella 1: Confronto tra codifica binaria e codifica grigia.

corrispondente valore floating point x

$$x = x_{min} + \frac{x_{max} - x_{min}}{2^l - 1}(\sum_{i=1}^{l} g[i]^{l-i}) \tag{16}$$

dove x_{min} e x_{max} sono rispettivamente il minimo e massimo valore che può assumere x e $g[i]$ è l'i-esimo allele del genotipo g.

Esempio 5.1. Se x è codificato con 8 bit e puo assumere valori nell'intervallo $[x_{min}, x_{max}] = [0, 1]$, allora il numero binario 01100111=103 è decodificato nel seguente valore:

$$x = 0 + \frac{1 - 0}{2^8 - 1}103 \approx 0.404$$

Un'alternativa alla classica codifica binaria è rappresenta dalla codifica grigia (gray code). Il vantaggio di tale codifica è che a valori vicini nella rappresentazione floating point corrispondono stringhe vicine (nel senso della distanza di Hamming) nella rappresentazione binaria. Per esempio, nella codifica binaria classica, il numero 00011111=31 non è adiacente al numero 00100000=32 , nonostante i numeri 31 e 32 siano adiacenti nella rappresentazione intera. In tal modo se 32 è una soluzione migliore di 31, l'algoritmo genetico deve cambiare 6 bit per ottenerlo. La codifica grigia risolve il problema perchè interi vicini sono rappresentati da stringhe che differiscono di un solo bit. La tabella 1 confronta la codifica binaria classica e la codifica grigia su stringhe di 3 bit, mentre la figura 9 presenta una pseudoprocedura per la decodifica grigia di un genotipo.

L'operatore di crossover maggiormente usato nella codifica binaria è il crossover a n punti. La differenza rispetto al crossover classico (detto a punto singolo) è l'utilizzo di n punti di taglio. Un altro operatore frequentemente utilizzato con la codifica binaria è il crossover uniforme che consiste nello scambiare reciprocamente, in maniera casuale, i bit dei due genitori in ogni posizione della stringa.

L'operatore di mutazione maggiormente utilizzato rimane quello proposto da Holland (paragrafo 3.3).

```
decodifica grigia (genotipo g)
{
      int n1=0;
      int x=0
      per i da 0 a l-1
      Se(g[i]=1)
      {
            n1 = n1 +1
            x = x + (n1 mod 2)*2^(l-1-i)
      }
      restituisci x
}
```

Figura 9: Pseudoprocedura per la decodifica grigia.

5.2 Codifica basata su numeri reali

La codifica basata su numeri floating point è quella più naturale per i problemi di ottimizzazione di parametri reali, anche se in molti casi si preferisce comunque la classica codifica binaria. La struttura dati utilizzata è un vettore di lunghezza l dove ogni elemento è un numero reale. In questo caso ogni soluzione candidata rappresenta un punto nello spazio della ricerca e non è necessario prevedere funzioni di decodifica del genotipo. La rappresentazione basata su numeri reali non pone problemi particolari per l'operatore di crossover e quelli proposti per gli algoritmi genetici a codifica binaria possono essere riproposti senza particolari problemi. Lo stesso non vale per l'operatore di mutazione.

La maggior parte degli operatori di mutazione per genotipi a valori reali alterano i geni degli individui g addizionandogli gli elementi di un vettore $M = (m_1, \ldots, m_l)$

$$\mathbf{g}' = \mathbf{g} + M$$

dove gli elementi di M possono essere generati in svariati modi, per esempio tramite distribuzioni uniformi $U(-\alpha, \alpha)$ per cui ogni m_i è un valore scelto nell'intervallo $[-\alpha, \alpha]$ con eguale probabilità.

5.3 Codifica ad albero

La codifica ad albero è utilizzata soprattutto nell'evoluzione di programmi, dove, piuttosto che generare direttamente soluzioni di un dato problema di ottimizzazione, l'obbiettivo è quello di derivare algoritmi in grado di risolvere particolari problemi computazionali. Tale branca di applicazione degli AG è chiamata Programmazione Genetica (PG) [6].

La struttura dati utilizzata è un albero con nodi terminali, detti foglie, e nodi non terminali. I nodi terminali, dai quali discendono sottoalberi, possono essere costanti e variabili specifiche del problema, mentre i nodi non terminali possono essere funzioni come +, -, *, / e $\sqrt{}$ o strutture di controllo del tipo *if then else*. La figura 10 rappresenta

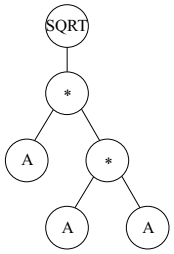

Figura 10: Esempio di codifica ad albero di un algoritmo che calcola l'espressione $\sqrt{A^3}$.

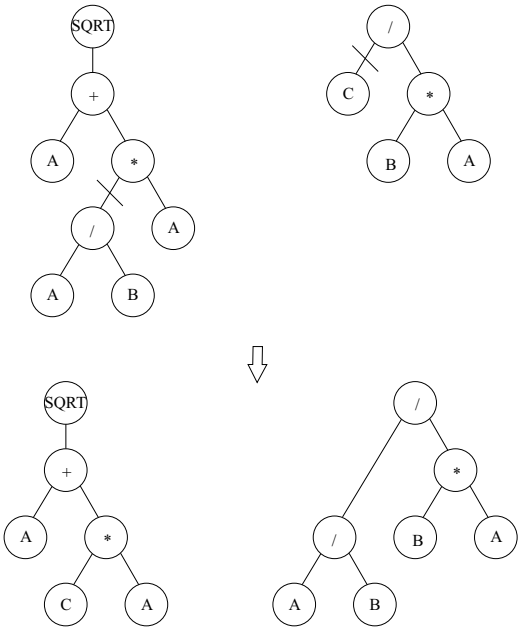

Figura 11: Esempio di crossover tra due alberi.

l'albero di un algoritmo che calcola l'espressione $\sqrt{A^3}$. L'albero si legge dal basso verso l'alto: prima si esegue il prodotto A*A, che si moltiplica ancora per A e, infine, se ne calcola la radice quadrata.

L'operatore di crossover e di mutazione presentano alcune differenze rispetto ai corrispondenti operatori degli AG.

L'incrocio consiste nella scelta di due punti di taglio, uno per ogni genitore, e nello scambio dei sottoalberi discendenti. La figura 11 mostra un esempio di crossover tra due alberi.

Come si evince dalla figura 11, per effetto dell'incrocio, le dimensioni dei programmi possono aumentare o diminuire.

L'operatore di mutazione, quando viene applicato, sostituisce sottoalberi con nuovi sottoalberi generati casualmente. Anche in questo caso le dimensioni dei programmi

possono aumentare o diminuire.

5.4 Metodi di selezione

La selezione è uno dei processi fondamentali di un AG poichè elimina gli individui con fitness bassa e crea una o più copie di individui con fitness alta da cui discendono gli individui della nuova popolazione. In generale, l'operatore di selezione può rimpiazzare l'intera popolazione, in tal caso si parla di AG generazionali, o soltanto una parte, caso in cui si parla di AG steady state. Negli AG generazionali può succedere che l'individuo migliore non venga selezionato per la riproduzione, mentre questo è generalmente considerata una buona idea. I modelli che garantiscono la selezione del miglior individuo sono detti elitistici, o k-elitistici se garantiscono la selezione dei migliori k.

5.4.1 Selezione proporzionale

L'originario modello di Holland impiegava la selezione proporzionale alla fitness (1). Tuttavia, gli individui con fitness alta e i loro discendenti si moltiplicano troppo velocemente (7) e impediscono di fatto all'algoritmo genetico di effettuare ulteriori significative esplorazioni [7].

5.4.2 Selezione in base al rango

La selezione in base al rango riduce significativamente la pressione selettiva rispetto alla selezione proporzionale. Gli individui della popolazione sono classificati in base alla fitness e i relativi valori attesi dipendono dalla posizione (rango) che occupano nella classifica [7]. Nel metodo di classificazione lineare proposto da Baker [1], ogni individuo è classificato in ordine crescente di idoneità, da 1 a n. L'utente scegli il valore atteso $Max \geq 0$ dell'individuo di rango n e il valore atteso di ogni individuo i della popolazione alla generazione t è

$$ExpVal(i,t) = Min + (Max - Min)\frac{rango(i,t) - 1}{n - 1} \tag{17}$$

dove Min è il valore atteso dell'individuo di rango 1. Dati i vincoli $Max \geq 0$ e $\sum i = 1^n ExpVal(i,t) = n$, affinchè la consistenza numerica della popolazione si mantenga inalterata tra una generazione e l'altra, è necessario che $1 \leq Max \leq 2$ e $Min = 2 - Max$. Baker suggerì di porre Max=1.1 e mostrò che questo schema dava risultati migliori di quello proporzionale all'idoneità in alcune applicazioni.

5.4.3 Selezione a torneo

La selezione a torneo è simile a quella basata sul rango per quanto riguarda la pressione selettiva, ma è più adatta per l'implementazione parallela [7]. Si scelgono a caso due individui e si genera un numero casuale $c \in [0,1]$. Se c risulta minore di un parametro $k \in [0,1]$ fissato, per esempio $k = 0.75$, allora si seleziona il più idoneo, altrimenti si

sceglie il meno idoneo. I due individui sono poi rimessi nella vecchia popolazione e possono essere scelti di nuovo.

Riferimenti bibliografici

[1] J. E. Baker. Adaptive selections methods for genetic algorithms. In J. J. Grefenstette (Eds), *Proceedings of the First Conference on Genetic Algorithms and Their Applications*, Erlbaum, 1999.

[2] E. Cantù-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2000.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* Wiley, 1966.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, second edition: mit press, 1992 edition, 1975.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[7] M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, 1996.

[8] I. Rechenberg. *Evolutionsstrategie: Optinierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Fromman-Holzboog (Struttgart), 1973.