Algoritmo genetico

Un **algoritmo genetico** è un algoritmo euristico utilizzato per tentare di risolvere problemi di ottimizzazione per i quali non si conoscono altri algoritmi efficienti di complessità lineare o polinomiale. L'aggettivo "genetico", ispirato al principio della selezione naturale ed evoluzione biologica teorizzato nel 1859 da Charles Darwin, deriva dal fatto che, al pari del modello evolutivo darwiniano che trova spiegazioni nella branca della biologia detta genetica, gli algoritmi genetici attuano dei meccanismi concettualmente simili a quelli dei processi biochimici scoperti da questa scienza.

In sintesi gli algoritmi genetici consistono in algoritmi che permettono di valutare diverse soluzioni di partenza (come se fossero diversi individui biologici) e che ricombinandole (analogamente alla riproduzione biologica sessuata), ed introducendo elementi di rumore (analogamente alle mutazioni genetiche casuali) producono nuove soluzioni (nuovi individui) che vengono valutate scegliendo le migliori (selezione ambientale) nel tentativo di convergere verso soluzioni "di ottimo". Ognuna di queste fasi di ricombinazione e selezione si può chiamare generazione come quelle degli esseri viventi. Nonostante questo utilizzo nell'ambito dell'ottimizzazione, data la natura intrinsecamente casuale dell'algoritmo genetico, non vi è modo di sapere a priori se sarà effettivamente in grado di trovare una soluzione accettabile al problema considerato. Se si otterrà un soddisfacente risultato non è detto che si capisca perché funzionato in quanto non è stato progettato da nessuno ma da una procedura casuale.

Gli algoritmi genetici rientrano nello studio dell'intelligenza artificiale e più in particolare nella branca della computazione evolutiva, vengono studiati e sviluppati all'interno del campo dell'intelligenza artificiale e delle tecniche di soft computing, ma trovano applicazione in un'ampia varietà di problemi afferenti a diversi contesti quali l'elettronica, la biologia e l'economia.

Storia

La nascita degli algoritmi genetici trova origine dalle prime teorizzazioni di Ingo Rechenberg che, per la prima volta, nel 1960, cominciò a parlare di "strategie evoluzionistiche" all'interno dell'informatica.

La vera prima creazione di un algoritmo genetico è tuttavia storicamente attribuita a John Henry Holland che, nel 1975, nel libro Adaptation in Natural and Artificial Systems pubblicò una serie di teorie e di tecniche tuttora di fondamentale importanza per lo studio e lo sviluppo della materia. Agli studi di Holland si deve infatti sia il teorema che assicura la convergenza degli algoritmi genetici verso soluzioni ottimali sia il cosiddetto teorema degli schemi, conosciuto anche come "teorema fondamentale degli algoritmi genetici". Quest'ultimo teorema fu originariamente pensato e dimostrato su ipotesi di codifica binaria ma nel 1991, Wright, l'ha estesa a casi di codifica con numeri reali dimostrando anche che una tale codifica è preferibile nel caso di problemi continui d'ottimizzazione[senza fonte].

Enormi contributi si devono anche a John Koza che nel 1992 inventò la programmazione genetica ossia l'applicazione degli algoritmi genetici alla produzione di software in grado di evolvere diventando capace di svolgere compiti che in origine non era in grado di svolgere.

Funzionamento

Prima dell'effettiva spiegazione del funzionamento degli algoritmi genetici, è necessario premettere che questi ereditano e riadattano dalla biologia alcune terminologie che vengono qui preventivamente presentate per una successiva maggiore chiarezza espositiva:

- *Cromosoma*: una delle soluzioni ad un problema considerato. Generalmente è codificata con un vettore di bit o di caratteri.
- *Popolazione*: insieme di soluzioni relative al problema considerato.
- *Gene*: parte di un cromosoma. Generalmente consiste in una o più parti del vettore di bit o caratteri che codificano il cromosoma.
- *Fitness*: grado di valutazione associato ad una soluzione. La valutazione avviene in base ad una funzione appositamente progettata detta *funzione di fitness*.
- *Crossover*: generazione di una nuova soluzione mescolando delle soluzioni esistenti.
- *Mutazione*: alterazione casuale di una soluzione.

Un tipico algoritmo genetico, nel corso della sua esecuzione, provvede a fare evolvere delle soluzioni secondo il seguente schema di base:

1. Generazione casuale della prima popolazione di soluzioni (cromosomi).
2. Applicazione della funzione di fitness alle soluzioni (cromosomi) appartenenti all'attuale popolazione.
3. Selezione delle soluzioni considerate migliori in base al risultato della funzione di fitness e della logica di selezione scelta.
4. Procedimento di crossover per generare delle soluzioni ibride a partire dalle soluzioni scelte al punto *3*.
5. Creazione di una nuova popolazione a partire dalle soluzioni identificate al punto *4*.
6. Riesecuzione della procedura a partire dal punto *2* ed utilizzando la nuova popolazione creata al punto *5*.

L'iterazione dei passi presentati permette l'evoluzione verso una soluzione ottimizzata del problema considerato.

Poiché questo algoritmo di base soffre del fatto che alcune soluzioni ottime potrebbero essere perse durante il corso dell'evoluzione e del fatto che l'evoluzione potrebbe ricadere e stagnare in "ottimi locali" spesso viene integrato con la tecnica dell'"elitarismo" e con quella delle mutazioni casuali. La prima consiste in un ulteriore passo precedente al punto 3 che copia nelle nuove popolazioni anche gli individui migliori della popolazione precedente, la seconda invece successiva al punto 4 introduce nelle soluzioni individuate delle occasionali mutazioni casuali in modo da permettere l'uscita da eventuali ricadute in ottimi locali.

Utile a comprendere il funzionamento di un GA è il concetto di "schema", introdotto da Holland (VEDI ALTRI MATERIALI).

La codifica

Come accennato le soluzioni al problema considerato, siano queste quelle casuali di partenza o quelle derivate da evoluzione, devono essere codificate con qualche tecnica.

Le codifiche più diffuse sono:

- *Codifica vettoriale binaria*: è la più diffusa, consiste in un vettore di *n* campi binari dove i valori 1 o 0 identificano delle caratteristiche elementari della soluzione. I vantaggi di questa tecnica risiedono nel fatto di essere semplice da implementare e da gestire durante l'intera evoluzione. Gli svantaggi consistono nelle difficoltà intrinseche della conversione delle soluzioni in questa codifica e dalle scarse possibilità rappresentative.
- *Codifica vettoriale reale*: come la codifica vettoriale binaria ma vengono utilizzati dei numeri reali.
- *Codifica vettoriale diretta*: consiste in una codifica vettoriale dove ogni campo contiene direttamente i valori relativi al problema. Il vantaggio è quello di una facile codifica, lo svantaggio risiede nella difficile gestione dell'algoritmo e nella difficile progettazione della funzione di fitness e dei processi di crossover e mutazione.
- *Codifica ad albero*: Ogni cromosoma è un albero di alcuni oggetti come ad esempio funzioni e comandi di un linguaggio di programmazione. In questo caso, per la sua particolare semantica e sintassi, viene spesso utilizzato il linguaggio di programmazione Lisp che semplifica notevolmente le operazioni di codifica.

La funzione di fitness

La funzione di fitness è quella che permette di associare ad ogni soluzione uno o più parametri legati al modo in cui quest'ultima risolve il problema considerato. Generalmente è associata alle prestazioni computazionali e quindi alle prestazioni temporali della soluzione.

La logica di selezione

A causa di complessi fenomeni di interazione non lineare, non è dato per scontato né che da due soluzioni promettenti ne nasca una terza più promettente né che da due soluzioni con valori di fitness basso ne venga generata una terza con valore di fitness più basso. Per ovviare a questi problemi, durante la scelta delle soluzioni candidate all'evoluzione, oltre che sul parametro ottenuto dalla funzione di fitness ci si basa anche su particolari tecniche di "selezione". Le più comuni sono:

- *Selezione a roulette*: la probabilità che una soluzione venga scelta per farla evolvere è direttamente proporzionale al valore restituito dalla funzione di fitness. Questa tecnica presenta dei problemi nel caso in cui ci siano delle grosse differenze di valori perché le soluzioni peggiori verrebbero selezionate troppo raramente.
- *Selezione per categoria*: simile alla selezione per roulette ma la valutazione è effettuata in maniera proporzionale alla somma del valore della funzione di fitness per ogni coppia possibile di soluzioni. Il problema presentato da questa tecnica di scelta è rappresentato dalla lentezza di convergenza nel caso in cui ci siano delle differenze troppo piccole tra coppie di soluzioni candidate.
- *Selezione a torneo*: utilizzata per prevenire patologie legate alla presenza accidentale di individui straordinari.

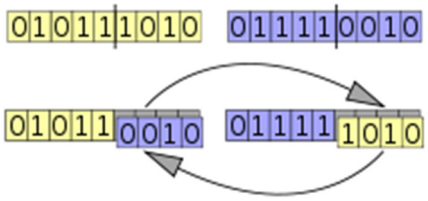
    A. Si seleziona casualmente un numero predefinito *s* di individui
    B. Si sceglie deterministicamente il migliore: sia esso Ind1
    C. Si iterano i passi (A) e (B) fino a generare N genitori
    D. Si applicano gli operatori genetici, creando N figli (la nuova popolazione)

    La conseguenza della selezione casuale inziale è dare la possibilità anche ad individui non fortissimi di riprodursi (nei tornei in cui non sono presenti gli individui migliori)

Il crossover

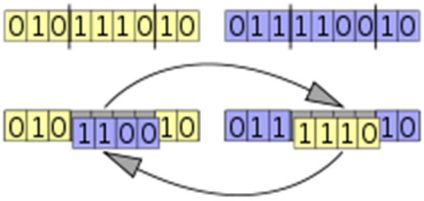
Per semplicità, durante la spiegazione del crossover, si farà riferimento alle codifiche vettoriali ma il procedimento per le codifiche ad albero è simile ed invece che essere applicato ai campi dei vettori viene applicato ai nodi dell'albero. In base ad un operatore stabilito inizialmente, alcune parti dei geni delle soluzioni candidate all'evoluzione vengono mescolate per ricavare nuove soluzioni.

Gli operatori più comunemente utilizzati sono:



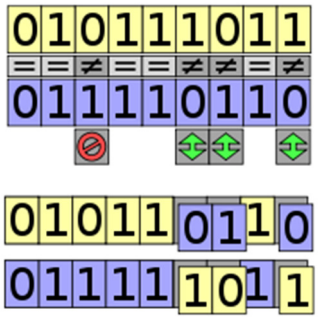
Crossover ad un punto

- Crossover ad un punto: consiste nel considerare due soluzioni adatte all'evoluzione e nel tagliare i loro vettori di codifica in un punto casuale o predefinito per ottenere due teste e due code. La prima nuova soluzione ottenuta sarà data dalla combinazione della testa della prima soluzione con la coda della seconda, mentre la seconda nuova soluzione sarà data dalla coda della prima soluzione con la testa della seconda.



Crossover a due punti

- Crossover a due punti: consiste nel considerare due soluzioni adatte all'evoluzione e nel tagliare i loro vettori di codifica in due punti predefiniti o casuali al fine di ottenere una testa, una parte centrale ed una coda dalla prima e dalla seconda soluzione. La prima nuova soluzione sarà data dalla testa e della coda della prima soluzione e dalla parte centrale della seconda soluzione. La seconda nuova soluzione sarà data dalla parte centrale della prima soluzione e dalla testa e dalla coda della seconda soluzione.



Crossover uniforme

- Crossover uniforme: consiste nello scambiare casualmente dei bit tra le soluzioni candidate all'evoluzione. Si segnala l'esistenza anche di crossover uniformi parziali ossia dei crossover uniformi dove lo scambio di bit è limitato ad una percentuale fissa o dinamica dei cromosomi candidati all'evoluzione.

Non è detto che il crossover debba avvenire ad ogni iterazione dell'algoritmo genetico. Generalmente la frequenza di crossover è regolata da un apposito parametro comunemente denominato “probabilità di crossover”.

La mutazione

La mutazione consiste nella modifica pseudocasuale di alcune parti dei geni in base a coefficienti definiti inizialmente. Queste modifiche alle volte sono utilizzate per migliorare il valore della funzione di fitness per la soluzione in questione e altre volte sono utilizzate per ampliare lo spazio di ricerca ed attuare la tecnica dell'elitarismo per non far ricadere l'evoluzione in ottimi locali. La frequenza con cui deve avvenire una mutazione è generalmente regolata da un apposito parametro comunemente denominato “probabilità di mutazione”.

Varianti

Algoritmo genetico multiobiettivo

Nel caso in cui si abbia più di un obiettivo da ottimizzare, è possibile utilizzare un algoritmo genetico multiobiettivo.

Sostanzialmente l'algoritmo funziona come quando va a perseguire un singolo obiettivo, quindi parte sempre da un certo numero di possibili soluzioni (la popolazione) e cerca di individuare, mediante diverse iterazioni, un certo numero di soluzioni ottimali, che si andranno a trovare su un fronte di Pareto. La diversità sta nel fatto che ora esistono due o più funzioni fitness da valutare.

Esempi didattici

Il problema del commesso viaggiatore

Il problema del commesso viaggiatore consiste nel riuscire a visitare almeno una volta tutte le città presenti in un elenco, sfruttando al meglio i collegamenti tra queste e percorrendo meno strada possibile.

- Codifica: probabilmente, il metodo più semplice per codificare questo problema è quello mediante l'utilizzo di vettori di interi. Il campo di un vettore associato ad un cromosoma andrebbe ad identificare in maniera univoca una delle città da visitare mentre il suo posizionamento andrebbe ad identificare l'ordine di visita. Ovviamente questa codifica deve rispettare dei vincoli legati alla natura del problema quali la presenza di una città almeno una volta in ogni cromosoma.
- Fitness: la funzione di fitness dovrebbe valutare i cromosomi in base alla distanza da percorrere per attuare il percorso codificato.
- Selezione, crossover e mutazione: per la metodica di selezione, crossover e mutazione sarebbe opportuno valutare differenti possibilità e probabilità a seconda della dimensione del problema. Per un problema di poche città, una particolare metodica potrebbe rivelarsi efficace ma non è detto che lo sia anche per un problema con molte città e viceversa. Il crossover e la mutazione devono essere ovviamente predisposti per mantenere la coerenza del problema considerato. In questo caso ad esempio, non sarebbe accettabile un crossover o una mutazione che vanno a rimuovere la visita di una città da una soluzione (VEDI LUCIDI)