



# Algoritmi genetici in Mathematica: soluzione del problema di Koza

Claudio Chiappetta

Università degli studi di Milano



# Il problema

Il problema

L'approccio

Ottimizzazione e risultati

- Parola bersaglio: "*universal*"
- Due insiemi di lettere: una pila e un insieme non ordinato
- Abbiamo un set di azioni
- Trovare una combinazione di azioni che formi la parola bersaglio sulla pila

Due metodi:

- Un individuo è un comando di Mathematica
- Un individuo è un albero (lista annidata di funzioni)

**Problema: Controllare l'esecuzione degli individui**

Si è scelto di usare dei nomi "finti" per le funzioni

# Generazione di individui casuali

Il problema

L'approccio

Ottimizzazione e risultati

- Si parte da  $EQ[\#1, \#2]$  &
- Una funzione ricorsiva inserisce come argomenti due comandi casuali, e chiama se stessa per riempire gli eventuali argomenti

# Esecuzione di un individuo

Il problema

L'approccio

Ottimizzazione e risultati

La struttura ad albero viene trasformata ricorsivamente in un espressione di Mathematica

- Una funzione trasforma un nodo in un comando di Mathematica, e ne determina gli argomenti applicando se stessa ai sottorami.
- Top-down, backtracking

# Esecuzione di un individuo

Il problema

L'approccio

Ottimizzazione e risultati

Nel comando con nomi inattivi, questi vengono sostituiti con le vere funzioni utilizzando `ReplaceAll`.

A questo punto l'espressione è immediatamente valutabile

## **Problema: la funzione DU**

È necessario valutare più di una volta gli argomenti. Si risolve utilizzando opportunamente `HoldAll`

- Funzioni che non premiano l'efficienza
  - ◆ Funzione standard
  - ◆ Funzione generosa
- Funzioni che premiano l'efficienza
  - ◆ Funzione di Koza
  - ◆ Variazione della funzione di Koza

# Crossover

Il problema

L'approccio

Ottimizzazione e risultati

- Presa una coppia di genitori, si stacca un ramo da ognuno dei due
- Si attacca su uno il ramo dell'altro



# Crossover: Individui del secondo tipo

Il problema

L'approccio

Ottimizzazione e risultati

- Estrarre un punto casuale in una lista di cui non si conosce lunghezza nè profondità

## Reservoir sampling

- L'algoritmo prende in ingresso un nodo dell'albero
- Lo mette nella riserva con probabilità  $1/passi$ , dove *passi* è il numero corrente di passi
- Chiama se stesso sui sottorami del nodo corrente
- Dopo che ha processato l'ultima foglia ritorna la riserva
- Top-down, backtracking
- $O(n)$ , dove  $n$  è il numero di nodi

# Crossover: Individui del secondo tipo

Il problema

L'approccio

Ottimizzazione e risultati

- L'inserzione di un ramo in un punto casuale è più complicata dell'estrazione
- Cambiare la gestione della riserva

# Crossover: Individui del primo tipo

Il problema

L'approccio

Ottimizzazione e risultati

## Estrazione

- RandomChoice, Level
- Una riga di codice

## Inserzione

- RandomChoice, Level, Position e ReplacePart
- 3 righe di codice

- Crossover su  $N_{individui} - 2$  elementi
- Vengono mantenuti due individui con fitness massima
- Mutazione

- $p_c \sim 0.7$
- $p_m \lesssim 0.01$  (alta)

## Problema

Vengono generati individui con loop infiniti. Si usano i metodi utilizzati da Koza per ottenere programmi efficienti

- Programmi funzionanti con lunghezza arbitraria vengono trovati prima della generazione 30.
- Si ottengono programmi che terminano dopo un numero di passi finito solo utilizzando una variazione della fitness di Koza
- In questo caso, si trovano programmi funzionanti fra la generazione 30 e la 50; programmi ragionevolmente efficienti compaiono dopo la generazione 60
- Tempo di esecuzione altamente variabile, dell'ordine di  $1 \div 10$  ore.