Paradigma Genetico

Corso di Apprendimento Automatico
Laurea Magistrale in Informatica

Nicola Fanizzi

Dipartimento di Informatica
Università degli Studi di Bari

10 dicembre 2009

Sommario

- Computazione Evoluzionistica
- Prototipi di programmazione prototipale
  - Esempio: GABIL
- Schema Theorem
- Programmazione Genetica
- Apprendimento individuale ed evoluzione di popolazioni

Computazione evoluzionistica

1. Procedure di calcolo ispirate dall'**evoluzione biologica**
2. Procedure di ricerca che applicano in modo stocastico operatori di ricerca ad un insieme di punti nello spazio di ricerca

Un po' di storia

- **1957, Box & 1961, Bledsoe**: approcci computazionali evolution-based
- **1960s**: introduzione di ulteriori approcci evoluzionari
- **1965, 1973 Rechenberg - 1975, 1977, 1995 Schwefel**: strategie evoluzionarie per l'ottimizzazione di parametri numerici nella progettazione ingegneristica;
- **1966 Folgel, Owens e Walsh**: inventano la evolutionary programming per l'evoluzione di macchine a stati finiti (seguiti da Fogel e Atmar 1993);
- **1992 Koza**: introduce la Genetic Programming
- **1980 K. DeJong** assieme ai sui studenti all'University of Pittsburg sviluppa l'approccio di usare i GA per apprendere insiemi di regole (sistema GABIL);
- **1986 Holland**: propone una variante per l'apprednimento di insieme di regole con GA;

Evoluzione biologica I

Secondo **Lamarck** ed altri:

- La specie muta nel tempo

Darwin e **Wallace**:

- Variazione coerente ed ereditaria tra gli individui in una popolazione
- Selezione naturale degli individui più adattati

Mendel e la genetica:

- Meccanismo per l'ereditarietà dei tratti somatici
- mapping: genotipo $\rightarrow$ fenotipo

Evoluzione biologica II

La metafora

$$\mathcal{F} : X \rightarrow \mathbb{R}$$

- Modello: evoluzione darwiniana delle popolazioni
  sopravvivenza del più adatto
- Termini:
  - Individuo: elemento $x \in X$
  - Fitness di x: $\mathcal{F}(x)$
  - Popolazione: $P = \{x_1, \ldots, x_N\}$
  - Generazione: dalla popolazione P_t alla popolazione P_{t+1}

Ingredienti

1. rappresentazione delle ipotesi
2. schema algoritmo evolutivo
   - criterio di valutazione delle ipotesi
   - criterio di stop
3. operatori di variazione
4. criterio di selezione delle ipotesi

Rappresentazione delle ipotesi

Nel problema **PlayTennis**:
l'attributo *Outlook* ammette i valori: $\{Sunny, Overcast, Rain\}$
l'attributo *Wind* ammette i valori: $\{Strong, Weak\}$

Si rappresenta una condizione

$$(Outlook = Overcast \vee Rain) \wedge (Wind = Strong)$$

con

Outlook	*Wind*
011	10

Si rappresenta una regola come

$$\text{IF } Wind = Strong \text{ THEN } PlayTennis = yes$$

con stringhe di bit (tutte di pari lunghezza)

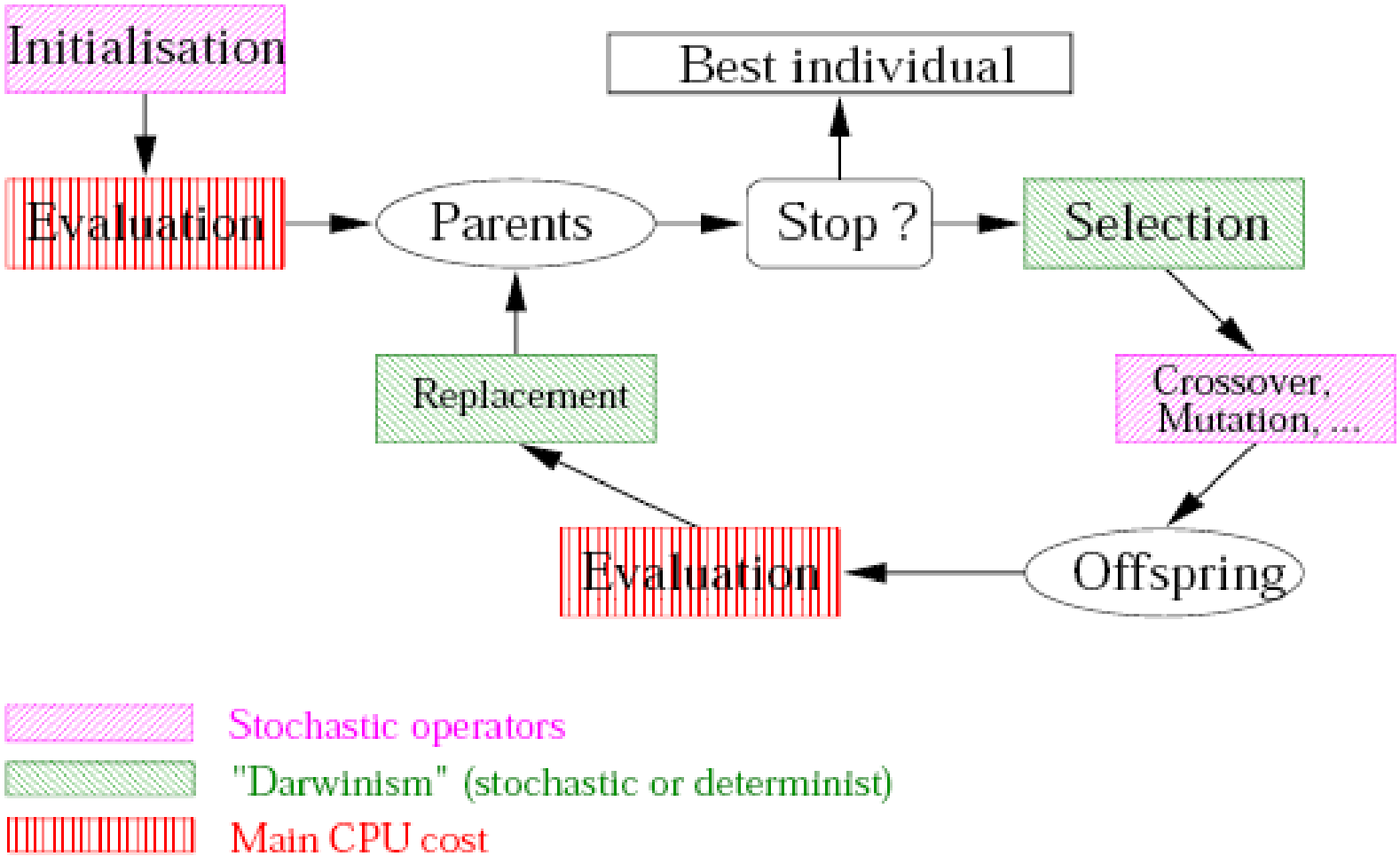
Outlook	*Wind*	*PlayTennis*
111	10	10

Inizializzazione della Popolazione

Selezionare $P_0 = \{x_1, \ldots, x_N\}$

- Campionamento uniforme:
  $H = \{0, 1\}^m$, $x_i^j = 0$ o 1 con probabilita' 0.5
  $H = [0, 1]^n$ $x_i^j =$ `random()`
  ...ma possono essere utilizzati altri criteri di uniformita'
- risultato di una precedente sessione di learning

Schema di Algoritmo Genetico I



Schema di Algoritmo Genetico II

GA($\mathcal{F}$:fitness; $\theta_{\mathcal{F}}$:soglia; p:# individui in una pop.;
r:% individui per il crossover; m:% individui per la mutazione)

- Inizializzazione popolazione: $P \leftarrow p$ ipotesi casuali
- Valutazione: **Per ogni** $h \in P$, calcolare $\mathcal{F}(h)$
- **Mentre** $\max_{h \in P} \mathcal{F}(h) < \theta_{\mathcal{F}}$
  1. Selezione: selezionare casualmente $(1 - r)p$ elementi di P da aggiungere a P_s [con prob. $\Pr(h_i) = \mathcal{F}(h_i) / \sum_{j=1}^{p} \mathcal{F}(h_j)$]
  2. Crossover: selezionare casualmente $rp/2$ coppie da P
     Per ogni coppia $\langle h_1, h_2 \rangle$:
     1. produrre due discendenti applicando l'operatore di crossover
     2. aggiungere tutti i discendenti a P_s
  3. Mutazione: invertire un bit a caso in mp elementi di P_s scelti a caso
  4. Aggiornamento: $P \leftarrow P_s$
  5. Valutazione: **Per ogni** $h \in P$, calcolare $\mathcal{F}(h)$
- Restituire l'ipotesi in P con la fitness maggiore: $argmax_{h \in P} \mathcal{F}(h)$

Fitness Function

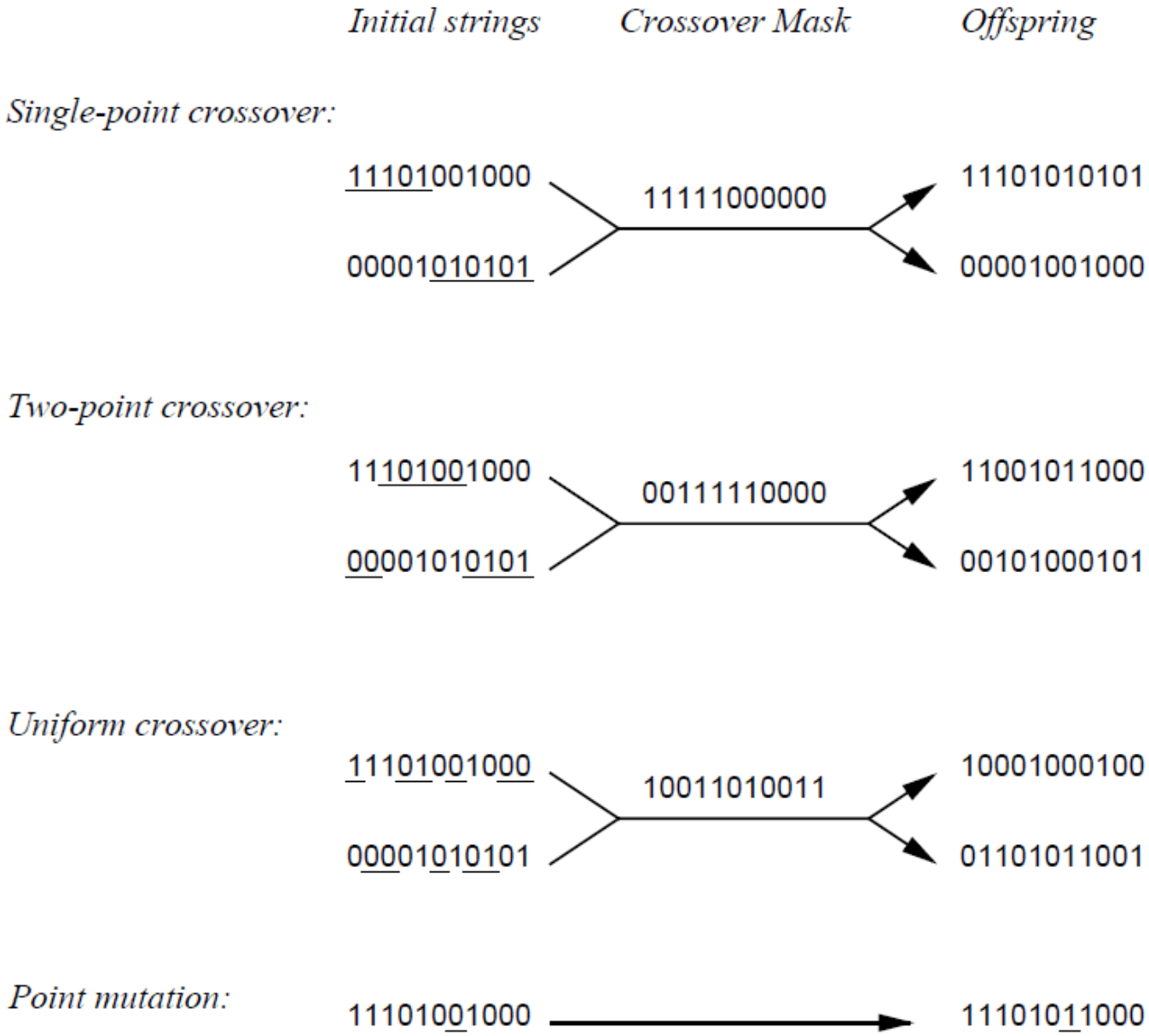
- passo più costoso
- definisce il criterio per valutare le ipotesi
- per regole di classificazione
  tipicamente ha una componente che calcola l'accuratezza della regola su un insieme di esempi di traning forniti
- È possibile utilizzare anche altri criteri
  es. complessità o livello di generalizzazione della regola

Operatori per algoritmi genetici I

Generazione di successori determinata da operatori che ricombinano e mutano individui selezionati nella popolazione corrente

- **Crossover Operator** $H \times H \rightarrow H \times H$
  Produce due nuovi discendenti da due stringhe genitori, copiando i bit selezionati da ogni genitore
  - bit in posizione i nei discendenti copiato dal bit in posizione i in uno dei due genitori
  - scelta del genitore che contribuisce al bit in posizione i determinata da una stringa aggiuntiva (crossover mask)
- **Point Mutation** $H \rightarrow H$
  inverte il valore di un bit

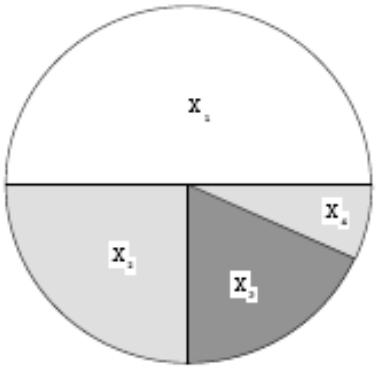
Operatori per algoritmi genetici II



Selezione delle ipotesi migliori I

- Selezione proporzionale alla fitness o roulette wheel:

$$Pr(\text{selezione di } x_i \propto \mathcal{F}(x_i)$$



$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

potrebbe portare ad un certo affollamento (*crowding*)

Selezione delle ipotesi migliori II

- Selezione a torneo:
  - Scegliere h_1, h_2 a caso con probabilità uniforme
  - con probabilità p, selezionare la migliore
- Selezione in base alla posizione:
  - Ordinare tutte le ipotesi in base alla fitness
  - La probabilità di selezione è proporzionale alla posizione in classifica

Il Sistema GABIL I

Impara insiemi (disgiunzioni) di regole proposizionali,
competitivo con C4.5 [DeJong et al., 1993]

Fitness:

$$Fitness(h) = (correct(h))^2$$

Rappresentazione:

$$\text{IF } a_1 = T \wedge a_2 = F \text{ THEN } c = T;\ \text{IF } a_2 = T \text{ THEN } c = F$$

rappresentata da

a_1	a_2	c		a_1	a_2	c
10	01	1		11	10	0

Il Sistema GABIL II

Operatori genetici: ???

- richiedono insieme di regole a lunghezza variabile
- richiedono solo ipotesi in forma di stringhe di bit ben formate

GABIL – Crossover con stringhe a lunghezza variabile

Si inizia con

	a_1	a_2	c	a_1	a_2	c
h_1 :	10	01	1	11	10	0
h_2 :	01	11	0	10	01	0

1. si scelgono punti di crossover per h_1, es., dopo i bit 1, 8
2. quindi si limitano i punti di h_2 ai soli che producono stringhe ben definite a livello semantico, es., $\langle 1, 3\rangle$, $\langle 1, 8\rangle$, $\langle 6, 8\rangle$.

se si sceglie $\langle 1, 3\rangle$, il risultato è

	a_1	a_2	c
h_3 :	11	10	0

	a_1	a_2	c	a_1	a_2	c	a_1	a_2	c
h_4 :	00	01	1	11	11	0	10	01	0

GABIL – estensioni

Si aggiungono nuovi operatori genetici,
applicati anche in modo probabilistico:

1. AggiungiAlternativa (AA):
   generalizza il vincolo su a_i mutando *uno* 0 in 1
2. EliminaCondizione (EC):
   generalizza il vincolo su a_i cambiando *ogni* 0 in 1

Inoltre, si aggiungono nuovi campi alla stringa di bit per determinare se consentire o meno tali operatori

a_1	a_2	c	a_1	a_2	c	AA	EC
01	11	0	10	01	0	1	0

Quindi anche la strategia di apprendimento evolve !

GABIL – Risultati

Le prestazioni di GABIL sono comparabili a quelle dei metodi simbolici di apprendimento di regole o alberi C4.5, ID5R, AQ14

Prestazioni medie su un insieme di 12 problemi sintetici:

- GABIL senza i due operatori: accuratezza del 92.1%
- GABIL con i due operatori: accuratezza del 95.2%
- i metodi simbolici hanno registrato accuratezze tra il 91.2% e il 96.6%

Crowding Problem I

Il crowding riguarda la tendenza degli individui migliori nella popolazione a riprodursi piu' velocemente

- nella popolazione copie dello stesso individuo, o individui molto simili tra loro

L'impatto negativo: ridzione della diversita' della popolazione, rendendo lenta la convergenza

Crowding Problem II

Approcci per ridurre il crowding

1. funzione di selezione che adotta la tournament o la rank selection
2. fitness strategy: la fitness viene diminuita in presenza di altri individui simili
3. si restringono i tipi di candidati alla ricombinazione per la discendenza
   - Es., permettendo che solo gli individui piu' simili si possano ricombinare, si incoraggia la formazione di cluster di individui simili, o "sottospecie" multiple nella popolazione
4. distribuire spazialmente gli individui e permettere che solo individui vicini si possano ricombinare

Schemi

Come caratterizzare l'evoluzione della popolazione negli algoritmi genetici ?
Holland (1975) fornisce una caratterizzazione in termini di schemi:
Schema = stringa contenente `0`, `1`, $*$ (non importa)

- Schema tipico:
  `10**0*`
- Istanze dello schema soprastante:
  `101101`, `100000`, ...

Si caratterizza la popolazione mediante il numero di istanze che rappresenta ogni possibile schema

- $m(s, t)$ = numero d'istanze dello schema s nella popolazione al tempo t

Teorema dello Schema – Selezione I

Il teorema dello schema descrive il valore atteso di $m(s, t+1)$ in termini di $m(s, t)$, di altre proprieta' dello schema, di popolazione, e dei parametri dell'algoritmo genetico

- $\bar{f}(t) =$ fitness media della popolazione al tempo t
- $m(s, t) =$ istanze dello schema s nella popolazione al tempo t
- $\hat{u}(s, t) =$ fitness media delle istanze di s al tempo t

Probabilità di selezionare h in un passo

$$\Pr(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} = \frac{f(h)}{n\bar{f}(t)}$$

Teorema dello Schema – Selezione II

Probabilità di selezionare un'istanza di s in un solo passo

$$\Pr(h \in s) = \sum_{h \in s \cap p_t} \frac{f(h)}{n\bar{f}(t)} = \frac{\hat{u}(s,t)}{n\bar{f}(t)} m(s,t)$$

Numero atteso di istanze di s dopo n selezioni

$$E[m(s,t+1)] = \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)$$

Il secondo passo segue dal fatto che per definizione

$$\hat{u}(s,t) = \frac{\sum_{h \in s \cap p_t} f(h)}{m(s,t)}$$

Teorema dello Schema – Selezione III

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

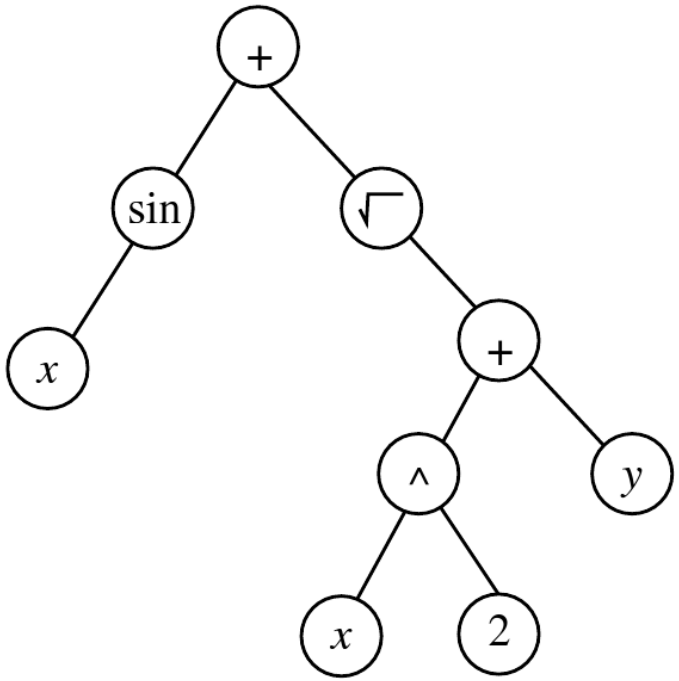
- $m(s, t)$ = istanze dello s nella pop. al tempo t
- $\bar{f}(t)$ = fitness media della pop. al tempo t
- $\hat{u}(s, t)$ = fitness media delle istanze di s al tempo t
- p_c = probabilità dell'operatore crossover single point
- p_m = probabilità dell'operatore di mutazione
- l = lunghezza della singola stringa di bit
- $o(s)$ = numero di bit definiti (non $*$) in s
- $d(s)$ = distanza tra i due bit definiti più a sinistra / destra in s

Programmazione genetica

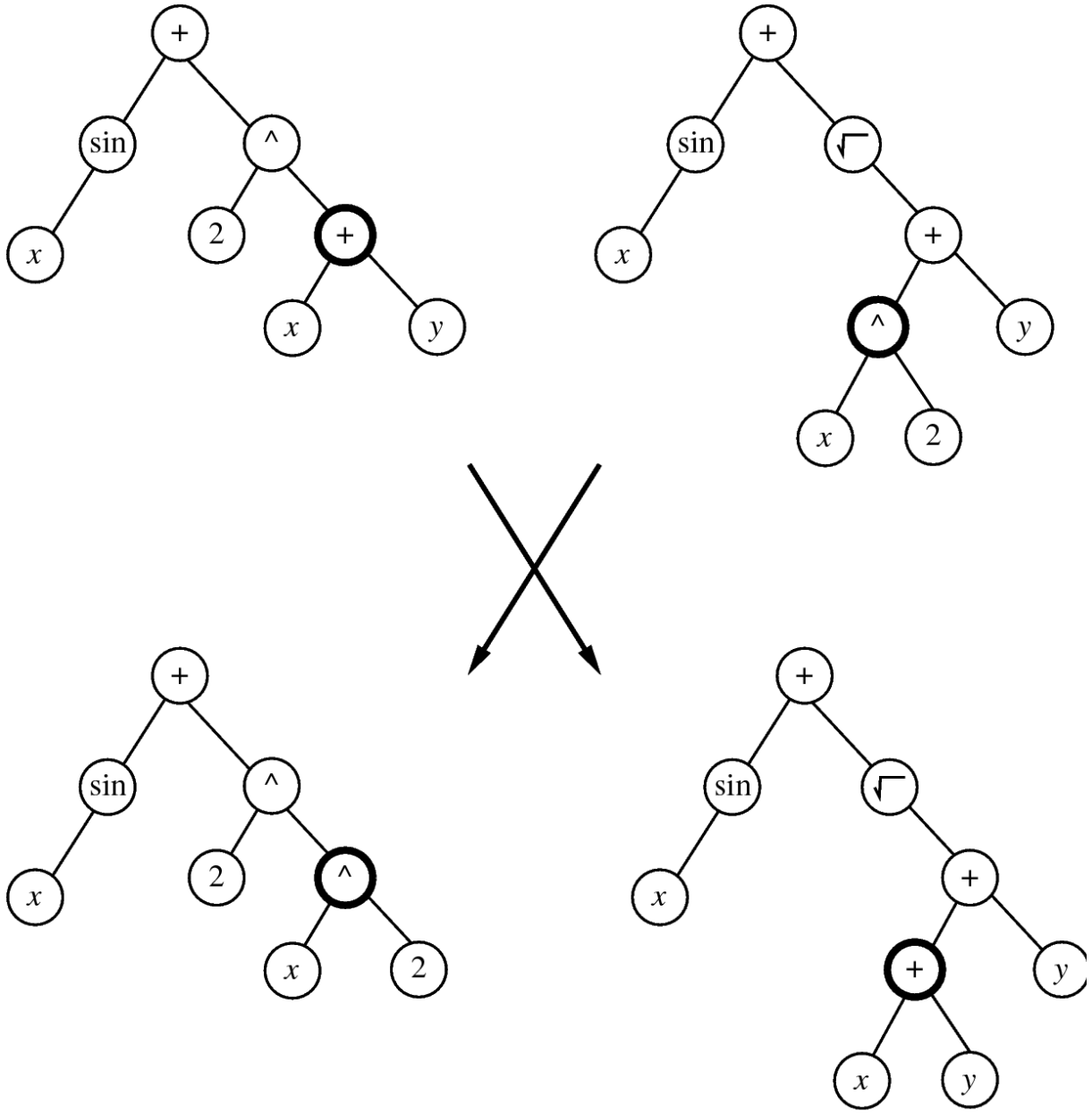
Nella **Programmazione Genetica** (GP) la popolazione è costituita di programmi rappresentati da programmi piuttosto che stringhe di bit [Koza, 1992]

I programmi manipolati dai GP sono tipicamente rapprsentati da alberi che corrispondono al parse tree del programma

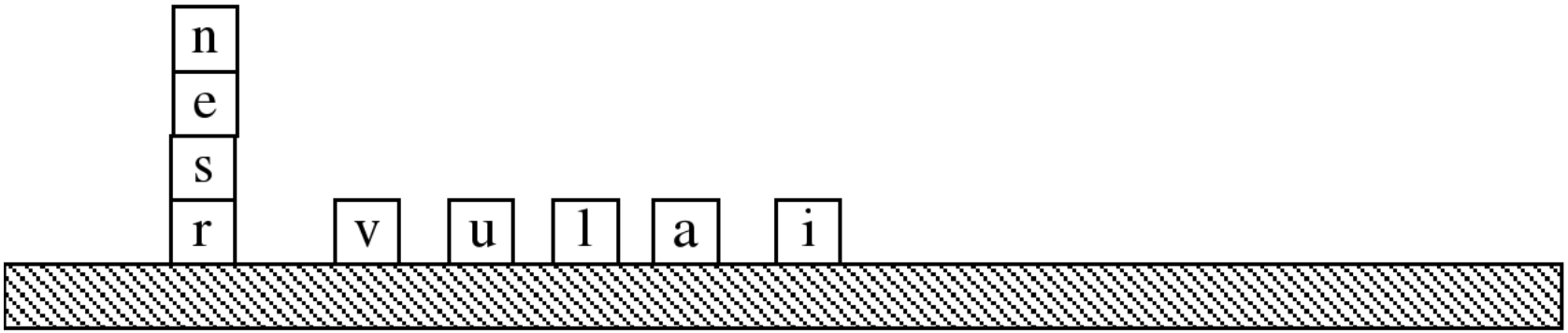
$$\sin(x) + \sqrt{x^2 + y}$$



Crossover



Problema dei blocchi I



Obiettivo: comporre la parola UNIVERSAL

Terminali:

- CS (“current stack”)
  nome del blocco al top dello stack, ossia *F*.
- TB (“top correct block”)
  nome del blocco corretto più alto sullo stack
- NN (“next necessary”)
  nome del prossimo blocco necessitato al di sopra di TB sullo stack

Problema dei blocchi II

Funzioni primitive:

- (MS x) (“move to stack”)
  se il blocco x è sul tavolo, muove x in cima allo stack e restituisce il valore T.
  Altrimenti, non fa nulla e restituisce il valore F.
- (MT x) (“move to table”)
  se il blocco x è sullo stack in qualche posizione, sposta il blocco in cima allo stack sulla tavola e restituisce T.
  Altrimenti, restituisce F.
- (EQ x y) (“equal”)
  restituisce T se x è uguale a y, e restituisce F altrimenti.
- (NOT x)
  restituisce T se $x = F$, altrimenti restituisce F
- (DU x y) (“do until”)
  esegue l’espressione x ripetutamente fino a quando l’espressione y non ritorni il valore T

Problema dei blocchi III

Programma appreso
Addestrato per adattarsi a 166 problemi di test

Usando la popolazione di 300 programmi
trovata dopo 10 generazioni:

```
(EQ (DU (MT CS)(NOT CS)) (DU (MS NN)(NOT NN)) )
```

Programmazione Genetica – esempi

Esempio più interessante:
progetto di circuiti elettronici filtro

- Gli individui sono programmi che trasformano circuiti iniziali in circuiti finali, aggiungendo/sottraendo componenti e connessioni
- Si usano popolazioni di 640.000 unità
  il sistema gira su processori paralleli a 64 nodi
- Si scoprono circuiti competitivi con i migliori circuiti progettati da umani

Classificazione di immagini tramite GP

Fitness: basata su copertura ed accuratezza

Rappresentazione:

- Le primitive comprendono `Add`, `Sub`, `Mult`, `Div`, `Not`, `Max`, `Min`, `Read`, `Write`, `If-Then-Else`, `Either`, `Pixel`, `Least`, `Most`, `Ave`, `Variance`, `Difference`, `Mini`, `Library`
  - `Mini` si riferisce a una subroutine locale co-evoluta separatamente
  - `Library` si riferisce a una subroutine globale di libreria (evoluta selezionando le `Mini` piï¿½ utili)

Operatori genetici:

- Crossover, mutazione
- Si creano mating pools e si usa la riproduzione proporzionale alla posizione in una graduatoria

[Teller and Veloso, 1994]

Evoluzione Biologica

Lamark (XIX secolo)

- si credeva che il patrimonio genetico individuale fosse alterato dall'esperienza nel corso della vita
- ma oggi ci sono prove del contrario

Qual è l'impatto dell'apprendimento individuale sull'evoluzione popolazione ?

Effetto Baldwin I

Si assuma

- L'apprendimento individuale non ha effetti diretti sul DNA dell'individuo
- ma l'abilità di apprendere riduce il bisogno di *cablare* i tratti genetici nel DNA

allora

- l'abilità degli individui di imparare supporterà i più diversi pool di geni
  - poichè l'apprendimento consente individui con individui con vari tratti genetici di successo cablati
- più disparati pool di geni sopporteranno un'evoluzione più veloce del pool

→ l'apprendimento individuale (indirettamente) incrementa il tasso di evoluzione

Effetto Baldwin II

Esempio plausibile:

1. Un nuovo predatore compare nell'ambiente
2. Gli individui che sanno imparare (ad evitarlo) verranno selezionati
3. Aumentare il numero degli individui che apprendono supporta pool di geni più diversificati
4. risultando in una evoluzione più veloce,
   a volte anche in nuovi tratti non appresi
   come ad es. la paura istintiva del predatore

Esperimenti sull'effetto Baldwin

Si fanno evolvere semplici reti neurali:

- Alcuni pesi nelle reti sono fissati per tutta la durata del ciclo di vita, altri sono addestrabili
- Il patrimonio genetico determina quali sono prefissati, e i loro valori

[Hinton and Nowlan, 1987]

Risultati

- Senza individui che apprendono, la popolazione non riuscirebbe a migliorare nel tempo
- Quando l'apprendimento individuale viene consentito
  - prime generazioni: la popolazione conteneva molti individuali con molti pesi addestrabili
  - generazioni finali: fitness più alta, mentre il numero dei pesi addestrabili decresceva

Sommario: programmazione evoluzionistica

- Si conduce una ricerca hill-climbing, casuale, parallela in H
- Si affronta l'apprendimento come un problema di ottimizzazione (si ottimizza la fitness)
- Caratteristica interessante:
  la valutazione della fitness può essere molto indiretta
  - si consideri l'imparare insiemi di regole di decisione multistep
  - assenza del problema di assegnazione di credito/colpa a passi individuali

Fonti

- T. M. Mitchell Machine Learning McGraw Hill

Approfondimenti

Testi

- Mitchell 1996 e Goldberg 1989: testi su GA
- Forret 1993: panoramica degli aspetti tecnici dei GA
- Goldberg 1994: panoramica delle recenti applicazioni degli GA
- Koza 1992: monografia sulla programmazione genetica

Conferenze

- International Conference on Genetic Algorithms
- Conference on Simulation of Adaptive Behavior
- International Conference on Artificial Neural Networks and Genetic Algorithms
- IEEE International Conference on Evolutionary Computation
- Evolutionary Computation Journal

Bibliografia I


DeJong, K. A., Spears, W. M., and Gordon, D. F. (1993).
Using genetic algorithms for concept learning.
Machine Learning, 13:161–188.


Hinton, G. and Nowlan, S. J. (1987).
How learning can guide evolution.
Complex Systems, 1:495–502.


Koza, J. (1992).
Genetic programming: On the programming of computers by means of natural selection.
MIT Press, Cambridge, MA.


Teller, A. and Veloso, M. (1994).
Pado: A new learning architecture for object recognition.
In *Symbolic visual learning*, pages 81–116. Oxford University Press.