

Intelligenza Artificiale II

Calcolo evolutivo

Elementi di teoria

Marco Piastra

Varianti del calcolo evolutivo

- Rappresentazione degli individui
 - Vettori di bit, di interi, di numeri reali
 - Grafi
 - Alberi
 - Dimensione fissa, limitata o variabile
- Operatori genetici
 - Mutazione, *crossover*
 - Operatori speciali (dipendenti dalla rappresentazione)
- Metodi di selezione
 - Roulette (fitness proportionate)*
 - Tournament*
- Processo evolutivo
 - Generation-based*
 - Steady state*
- Fitness
 - Scalare
 - Multi-valore (multi-obiettivo)

Evoluzione di stringhe di bit: schemi e iperpiani

- Spazi di ricerca

Si consideri una popolazione di stringhe binarie di lunghezza tre

Lo spazio di ricerca (tutti i possibili individui) può essere rappresentato come un **cubo**

- Schemi

Lo schema 1^{**} descrive tutti gli individui aventi un 1 al primo posto

Vale a dire, un piano del cubo

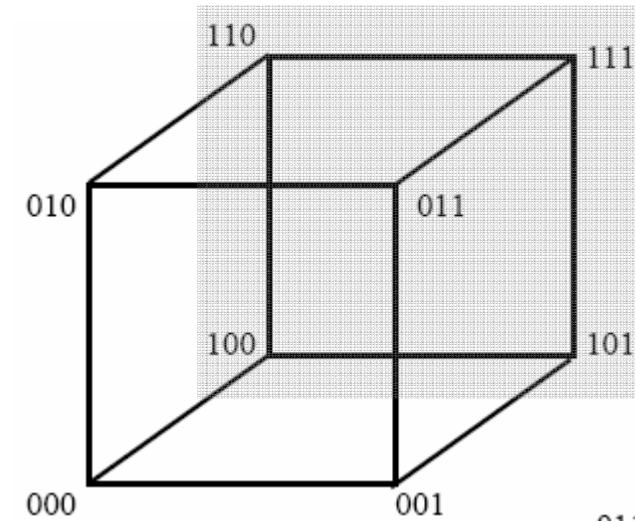
Il piano complementare è 0^{**}

- Ordine degli schemi e generalizzazione

1^{**} descrive un piano di ordine 1, mentre $*01$ descrive un piano di ordine 2
(ordine = numero di bit definiti)

Generalizzando:

- Lo spazio delle stringhe binarie (di lunghezza n) è un **ipercubo** (di ordine n)
- Ogni schema descrive un **iperpiano** (di ordine pari al numero di bit definiti)



Schemi, campionamento e fitness



■ Schemi e campionamento

Una popolazione contiene un campione di ciascuno schema

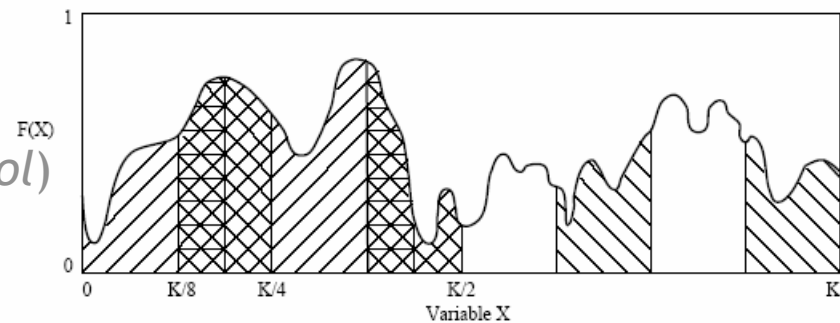
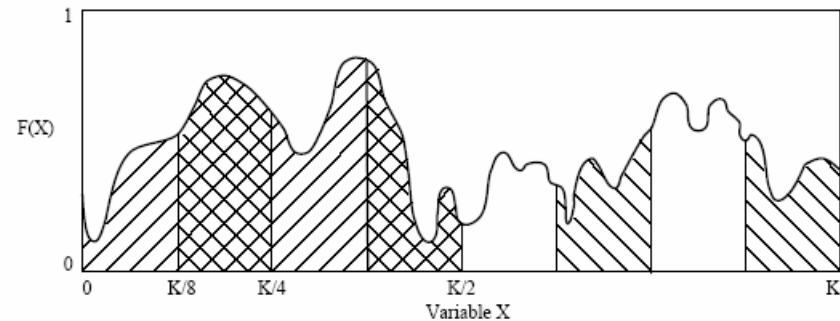
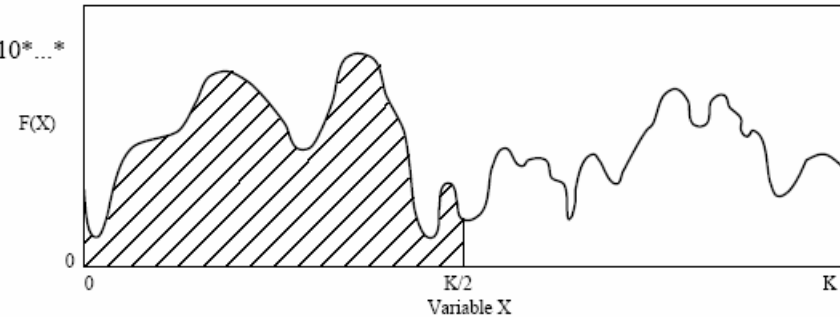
Si assume che la ricerca avvenga su più iverpiani in parallelo

■ Fitness di uno schema

E' il valor medio delle *fitness* del campione (i.e. tutti gli individui appartenenti all'iperpiano)

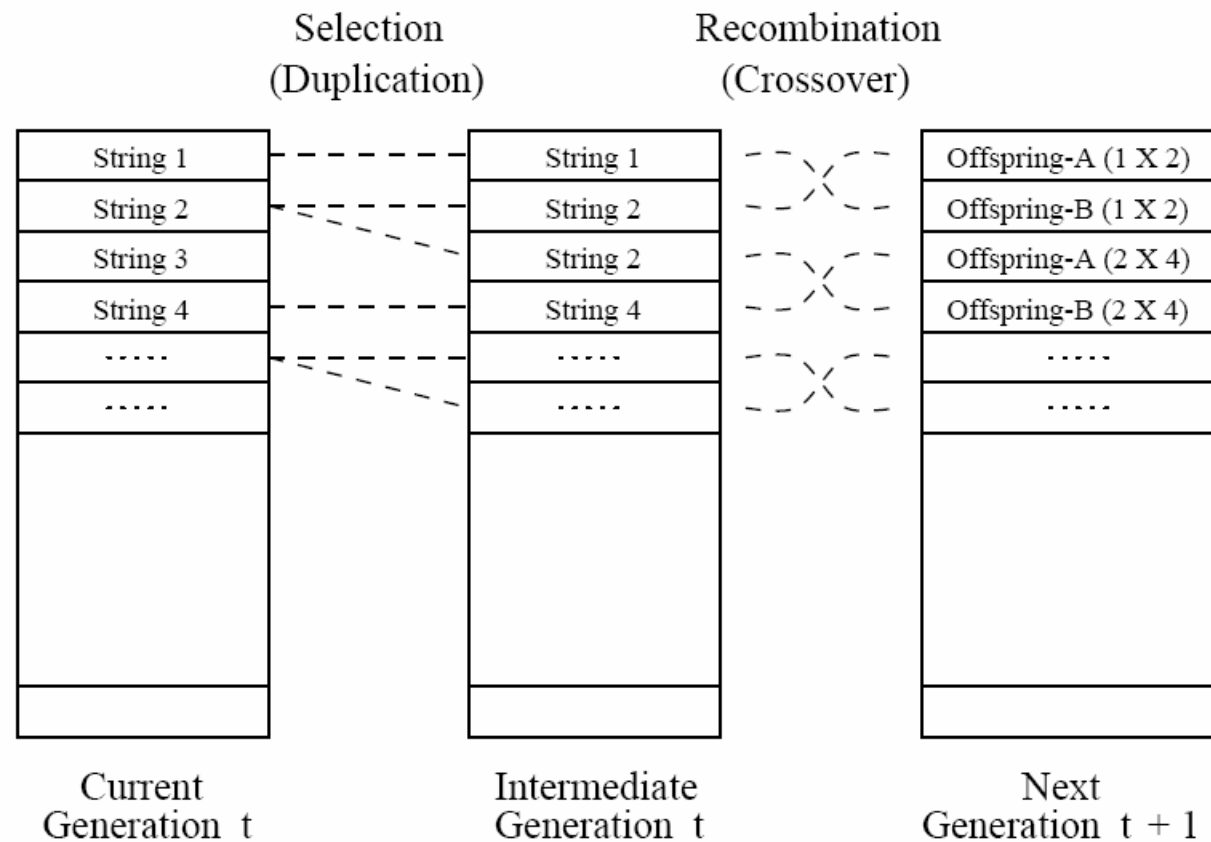
■ Selezione *fitness proportionate*

La strategia di selezione premia (con una maggiore presenza nel *mating pool*) i campioni a *fitness* più elevata



Mating pool

Serve a descrivere i risultati che seguono



Il *mating pool* è l'insieme degli individui selezionati per la ricombinazione

Campioni e selezione

■ Esempio

Costruzione di un *mating pool*

Popolazione casuale
di 21 individui

Mean

fitness media dello schema

Count

campioni nella popolazione

Expect

attesi nel *mating pool*

Obs

campioni nel *mating pool*

| Schemata and Fitness Values | | | | | | | | | |
|-----------------------------|-------|-------|--------|-----|----------|-------|-------|--------|-----|
| Schema | Mean | Count | Expect | Obs | Schema | Mean | Count | Expect | Obs |
| 101*...* | 1.70 | 2 | 3.4 | 3 | *0**...* | 0.991 | 11 | 10.9 | 9 |
| 111*...* | 1.70 | 2 | 3.4 | 4 | 00**...* | 0.967 | 6 | 5.8 | 4 |
| 1*1*...* | 1.70 | 4 | 6.8 | 7 | 0***...* | 0.933 | 12 | 11.2 | 10 |
| *01*...* | 1.38 | 5 | 6.9 | 6 | 011*...* | 0.900 | 3 | 2.7 | 4 |
| **1*...* | 1.30 | 10 | 13.0 | 14 | 010*...* | 0.900 | 3 | 2.7 | 2 |
| *11*...* | 1.22 | 5 | 6.1 | 8 | 01**...* | 0.900 | 6 | 5.4 | 6 |
| 11**...* | 1.175 | 4 | 4.7 | 6 | 0*0*...* | 0.833 | 6 | 5.0 | 3 |
| 001*...* | 1.166 | 3 | 3.5 | 3 | *10*...* | 0.800 | 5 | 4.0 | 4 |
| 1***...* | 1.089 | 9 | 9.8 | 11 | 000*...* | 0.767 | 3 | 2.3 | 1 |
| 0*1*...* | 1.033 | 6 | 6.2 | 7 | **0*...* | 0.727 | 11 | 8.0 | 7 |
| 10**...* | 1.020 | 5 | 5.1 | 5 | *00*...* | 0.667 | 6 | 4.0 | 3 |
| *1**...* | 1.010 | 10 | 10.1 | 12 | 110*...* | 0.650 | 2 | 1.3 | 2 |
| ****...* | 1.000 | 21 | 21.0 | 21 | 1*0*...* | 0.600 | 5 | 3.0 | 4 |
| | | | | | 100*...* | 0.566 | 3 | 1.70 | 2 |

■ In generale

$$M(H, t + intermediate) = M(H, t) \frac{f(H, t)}{\bar{f}}$$

$M(H, t)$ è la dimensione del campione dell'iperpiano H alla generazione t

$t + intermediate$ è l'istante intermedio tra due generazioni

Dopo la selezione del *mating pool*, prima di *crossover* e mutazione

$f(H, t)$ è la *fitness* media del campione di H alla generazione t

\bar{f} è la *fitness* media del campione

Crossover, mutazione e schemi

- *Crossover* e vulnerabilità degli schemi

Un *crossover* che 'taglia a metà' uno schema lo sconvolge:
schemi più lunghi sono dunque più vulnerabili

La *defining length* $\Delta(H)$ di uno schema è la distanza massima tra i bit definiti

$\Delta(H)$ di ****10****** è 1

$\Delta(H)$ di ****10*0**** è 3

$\Delta(H)$ di **1*10**0*** è 6

La probabilità che il *crossover* 'tagli' uno schema è quindi $p_c \frac{\Delta(H)}{L-1}$,
dove L è la lunghezza degli individui e p_c la probabilità
di occorrenza del *crossover*

- Mutazione e schemi

Detto $o(H)$ l'ordine dello schema H (il numero di bit predefiniti)

La probabilità che lo schema sopravviva intatto è $(1 - p_m)^{o(H)}$

dove p_m è la probabilità che uno dei bit definiti venga mutato

Schema Theorem (Holland, 1975)

■ Probabilità e generazioni

Definita come $P(H,t)$ la frequenza relativa dei campioni di H nella generazione t (i.e. una probabilità)

Il teorema stabilisce una relazione tra $P(H,t)$ e $P(H,t+1)$

$$P(H,t+1) \geq P(H,t) \frac{f(H,t)}{\bar{f}} \left[1 - p_c \frac{\Delta(H)}{L-1} \left(1 - P(H,t) \frac{f(H,t)}{\bar{f}} \right) \right] (1 - p_m)^{o(H)}$$

| Frequenza relativa di campioni di H alla generazione $t+1$
 |
 |
 | Probabilità di sopravvivenza a *crossover* e mutazione
 |
 | Proporzione attesa per la selezione
 |
 | Frequenza relativa di campioni di H alla generazione t

Caratteristiche del calcolo evolutivo

▪ Processo evolutivo

Convergente (nel senso di crescita degli schemi a *fitness* più alta)

(data una ragionevole scelta di p_c e p_m)

Non c'è garanzia di convergenza verso un ottimo globale

Come metodo di ottimizzazione

Black box:

le operazioni sul genotipo prescindono dalle caratteristiche della funzione di fitness

Approssimato: manca la garanzia di convergenza ad un ottimo (locale o globale)

Anytime: in qualsiasi momento si può ottenere una soluzione approssimata

Robusto: poco influenzato dai massimi locali

▪ Operatori genetici ed individui

Gli schemi **corti** e **compatti** 'sopravvivono' meglio

Minore probabilità di 'taglio' da parte del crossover

Minore probabilità di sconvolgimento da parte della mutazione

Gli schemi devono avere una lunghezza sufficiente

Che permetta loro di esprimere un carattere significativo

Building-block hypothesis

- ***Dallo Schema Theorem***

I blocchi compatti e più corti hanno maggiore probabilità di sopravvivere intatti al campionamento (*crossover*)

Aumentando quindi rapidamente la propria presenza nella popolazione

- ***Building-block hypothesis***

Il calcolo evolutivo è particolarmente efficiente su problemi la cui soluzione è decomponibile in blocchi (di qualsiasi natura)

Ciò spiegherebbe la grande dipendenza dei risultati dal metodo di codifica del genotipo

Riscontrabile nella pratica

Intuitivamente vero nel caso del GP (programmazione a blocchi)

L'ipotesi si scontra con numerosi esempi controfattuali

Non vale quindi in generale

Ancora sullo *Schema Theorem*

Lo *Schema Theorem* descrive l'occorrenza di campioni in due generazioni successive
Non descrive la tendenza a lungo termine del processo

▪ Limiti di popolazione

Si consideri una popolazione di 100 stringhe di 8 bit, generate casualmente
In media si avranno:

50 campioni di ****1*******

25 campioni di ****10******

12.5 campioni di ****101*****

$\frac{100}{2^n}$ campioni di uno schema di ordine n

Il campionamento può diventare insufficiente per schemi più grandi

▪ Limiti di crescita dei campioni

Due schemi ***1******* e ****00****** potrebbero avere entrambi una *fitness* elevata

Ma sono in conflitto (terzo bit): i due campioni non possono crescere indefinitamente

La crescita simultanea dei campioni è possibile solo per schemi compatibili

No Free Lunch Theorem (Walpert & Macready, 1996)

- Le performance medie dei metodi *black-box* sono identiche
Si considerano tutte le possibili funzioni da ottimizzare

$$\sum_f P(d_m^y | f, m, a_1) = \sum_f P(d_m^y | f, m, a_2).$$

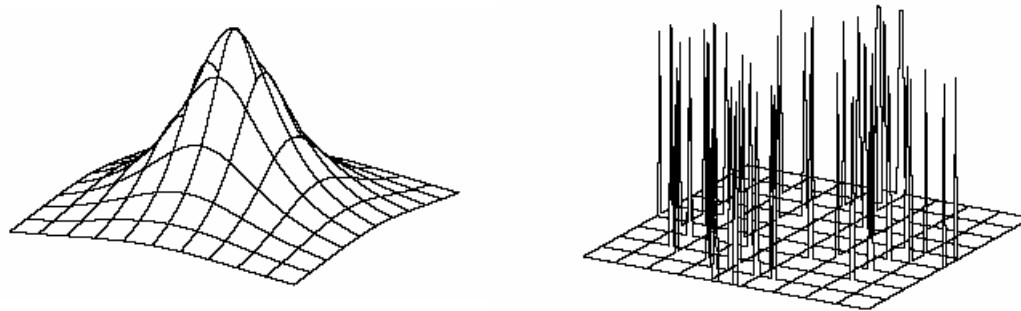
$P(d_m^y | f, m, a_1)$ è la probabilità che un algoritmo a_1 trovi un campione per cui la funzione f ha valore d_m^y in esattamente m passi
La sommatoria è estesa a tutte le possibili funzioni f

- Significato
Non esiste un metodo *black-box* che sia migliore in tutti i casi
La bontà del metodo è tipicamente legata alla classe di problemi

Spazio di ricerca del *Genetic Programming*

Piccole variazioni del genotipo (p.es. cambio di un'istruzione) provocano grandi variazioni del valore di fitness

Si parla di *ruggedness* (asprezza) del *fitness landscape* in un problema GP



■ Osservazioni

In un simile spazio di ricerca, si possono applicare solo metodi *black box*

Per il *No Free Lunch Theorem*, il calcolo evolutivo (ed il GP in particolare) è migliore della ricerca casuale solo in una determinata classe di problemi

Nella pratica, spesso lo è

Ma non esiste un soddisfacente inquadramento teorico di tale classe di problemi

Problemi tipici: *takeover*

- Perdita della diversità della popolazione

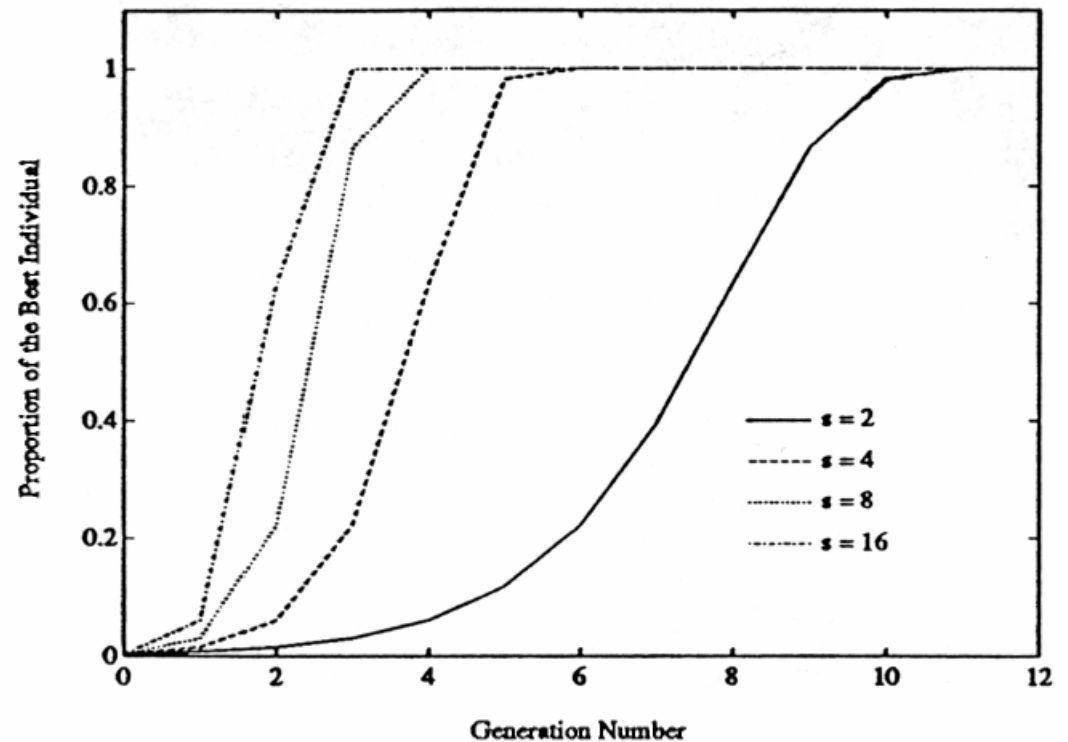
Un aumento eccessivo della pressione selettiva per accelerare il processo

p.es. aumento della dimensione della dimensione k
nella selezione a *tournament*

Causa il prevalere
di un singolo individuo

la popolazione perde
diversità

il processo evolutivo
si arresta



Problemi tipici: *bloating*

- Gigantismo degli individui (*bloating*)

Gli operatori genetici (p.es. *crossover*) non limitano le dimensioni dell'*offspring*

Individui più grandi (con parti ridondanti) preservano meglio la fitness

E` più alta la probabilità che gli operatori genetici alterino le parti ridondanti

Risultato: esplosione delle dimensioni degli individui

Fenomeno molto più marcato nel caso di processi non convergenti
(può essere un indizio ...)

Rimedi:

In un processo convergente, spesso il problema si risolve da solo

Uso della *parsimony pressure*:

la fitness penalizza gradualmente gli individui più grandi

Esempio: robot

- Un robot che si muove in un ambiente

L'obiettivo è evitare gli ostacoli

La logica di controllo utilizza un ciclo *sense-eval-act*:

Lettura dei sensori (*sense*)

Valuta le informazioni (*eval*)

Comandi agli attuatori (*act*)

L'obiettivo dell'evoluzione è una logica di controllo

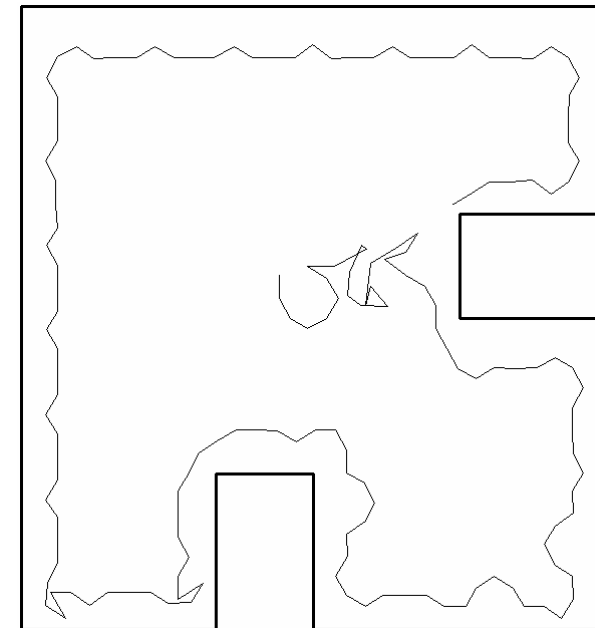
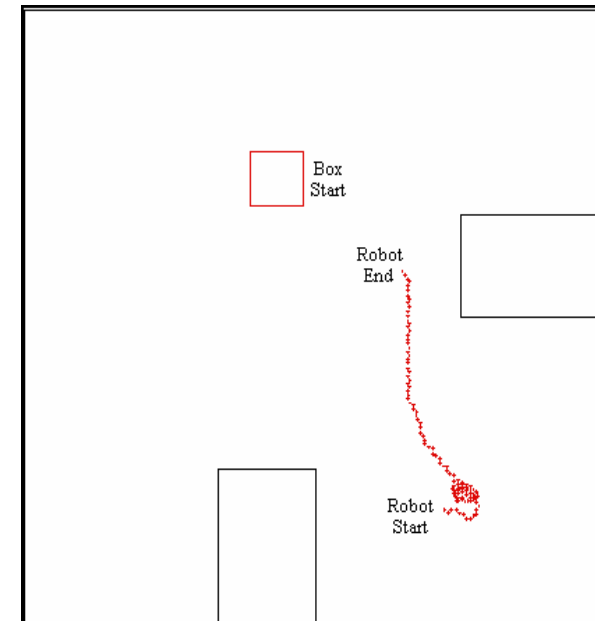
Esplorazione efficace e completa

Senza urtare gli ostacoli

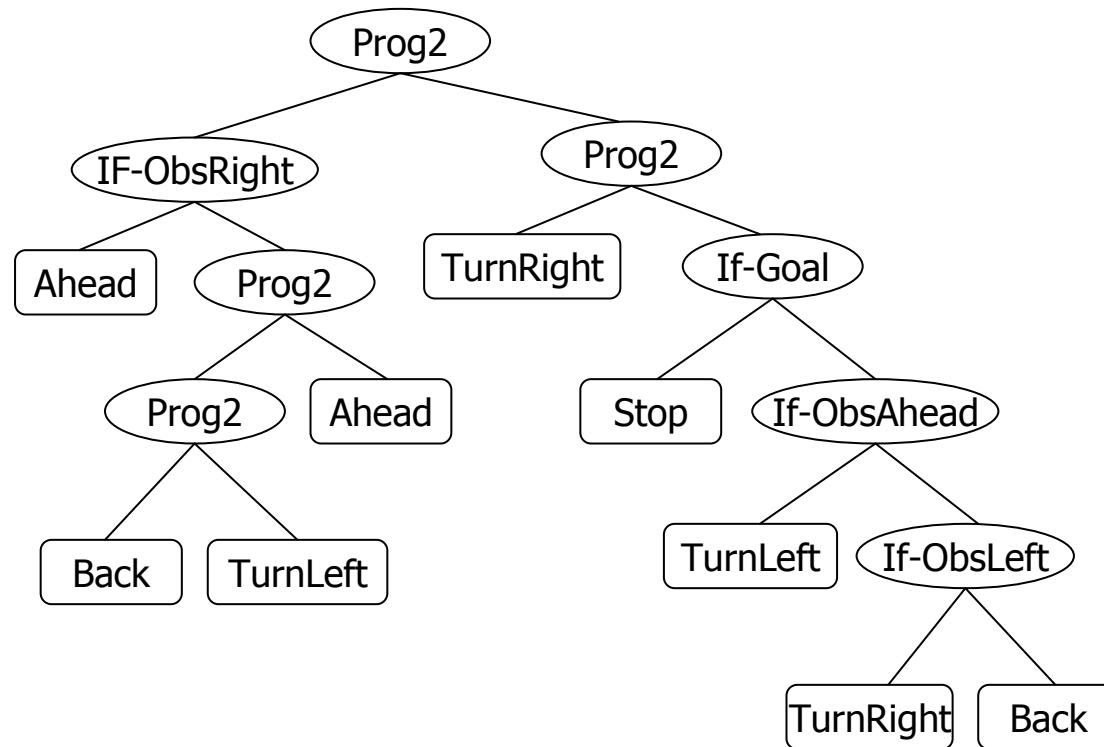
La *fitness* valuta il comportamento

+ Area esplorata, - Collisioni

La valutazione della fitness prende tempo ...



Robot: tipico individuo



Robot: osservazioni

- I nodi dell'albero sono *typeless*
Questo facilita la generazione incrementale
- Non c'è passaggio di valori
La struttura dei nodi descrive solo il flusso dell'esecuzione
I sotto-alberi non passano un valore alla loro radice
- Non si usa memoria
Ogni ciclo *sense-eval-act* è un episodio indipendente
- Problemi per l'estensione
Il passaggio dei valori tende a introdurre la gestione dei *tipi*
Si consideri il caso: IF <sense> THEN <eval> ELSE <act>
La gestione della memoria può non essere semplice
Si può incorporare nei nodi ...