UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica
Anno accademico 2021/2022

Fondamenti di Intelligenza Artificiale

Lezione 7 - Algoritmi di ricerca locale (II)

Algoritmi Genetici

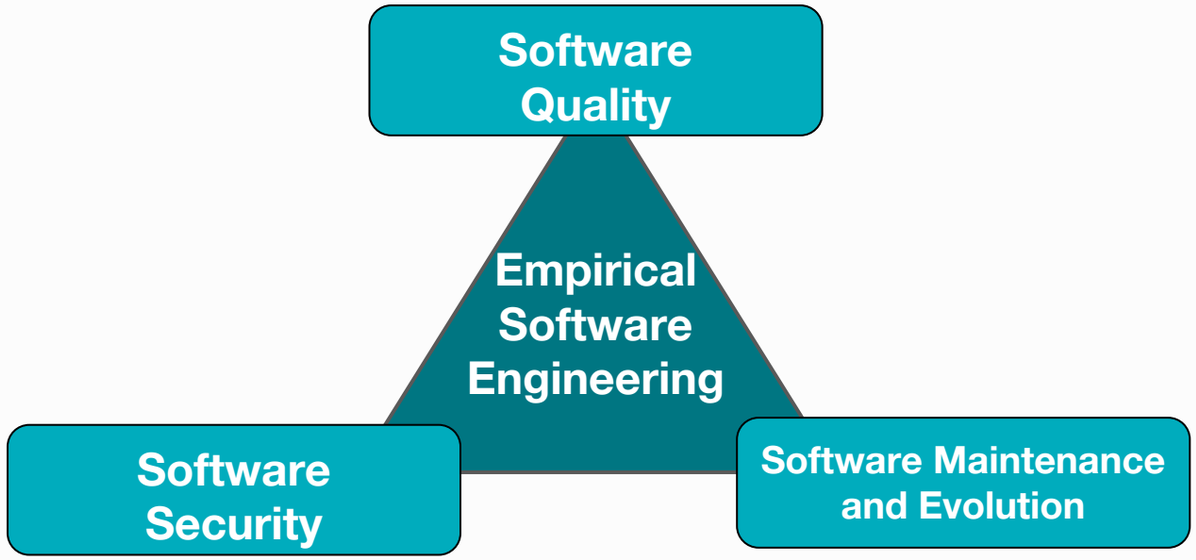
Today's Speaker

Emanuele Iannone
II Year PhD Student

https://emaiannone.github.io/

Emanuelelannone3

eiannone@unisa.it

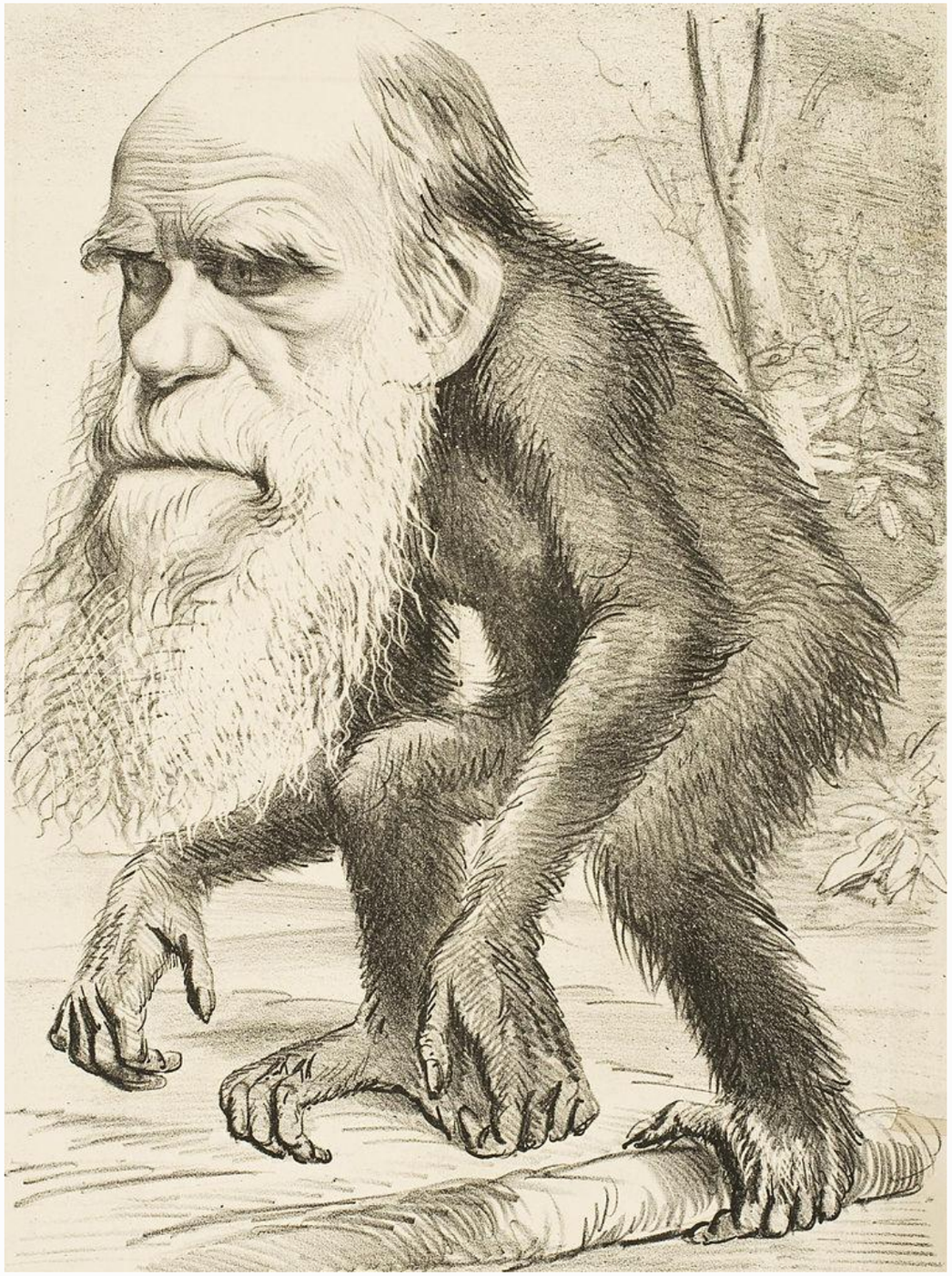


Algoritmi Genetici

Algoritmi Genetici

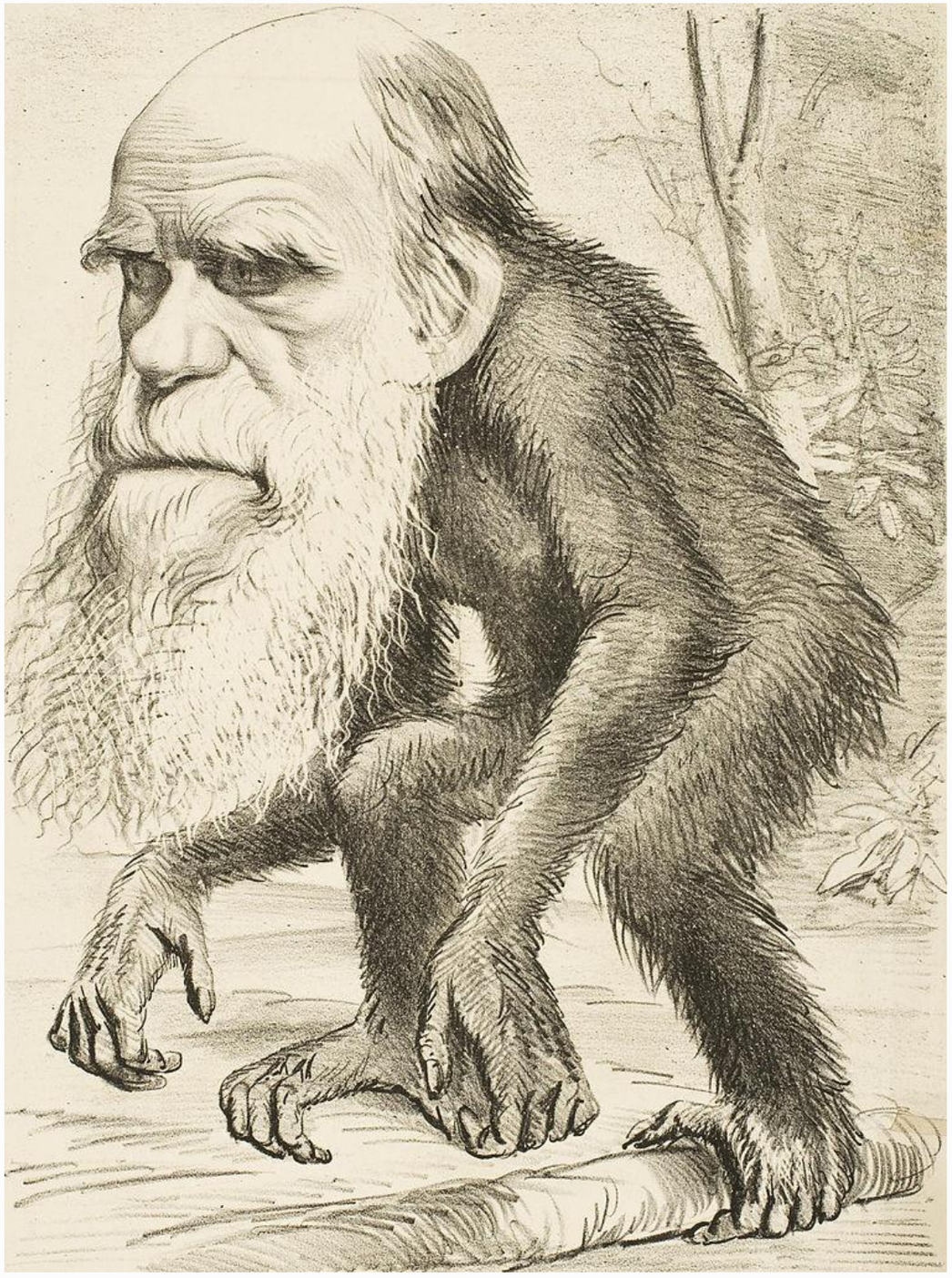
Algoritmi Genetici

Un po’ di contesto...



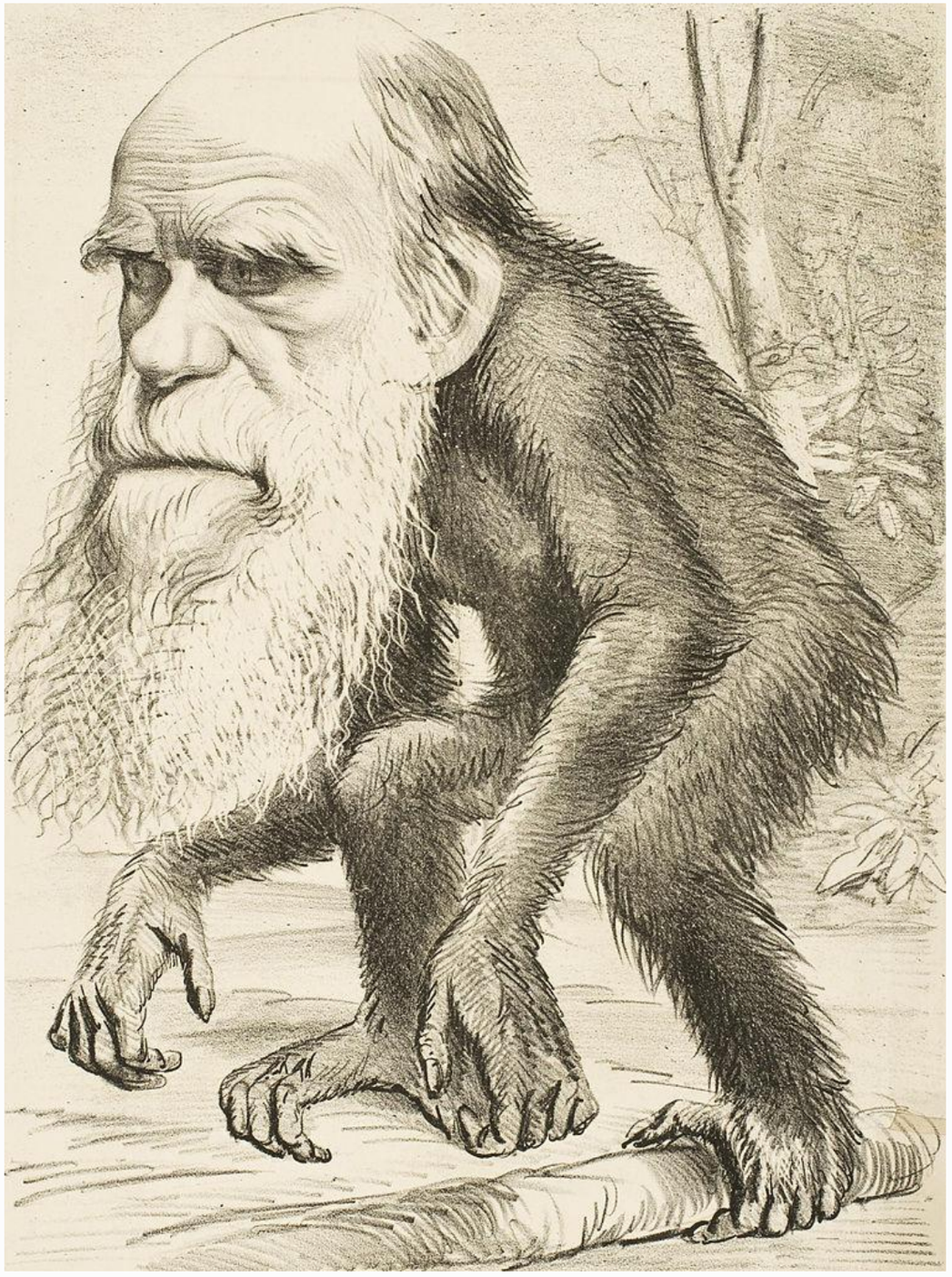
Algoritmi Genetici

Un po’ di contesto...



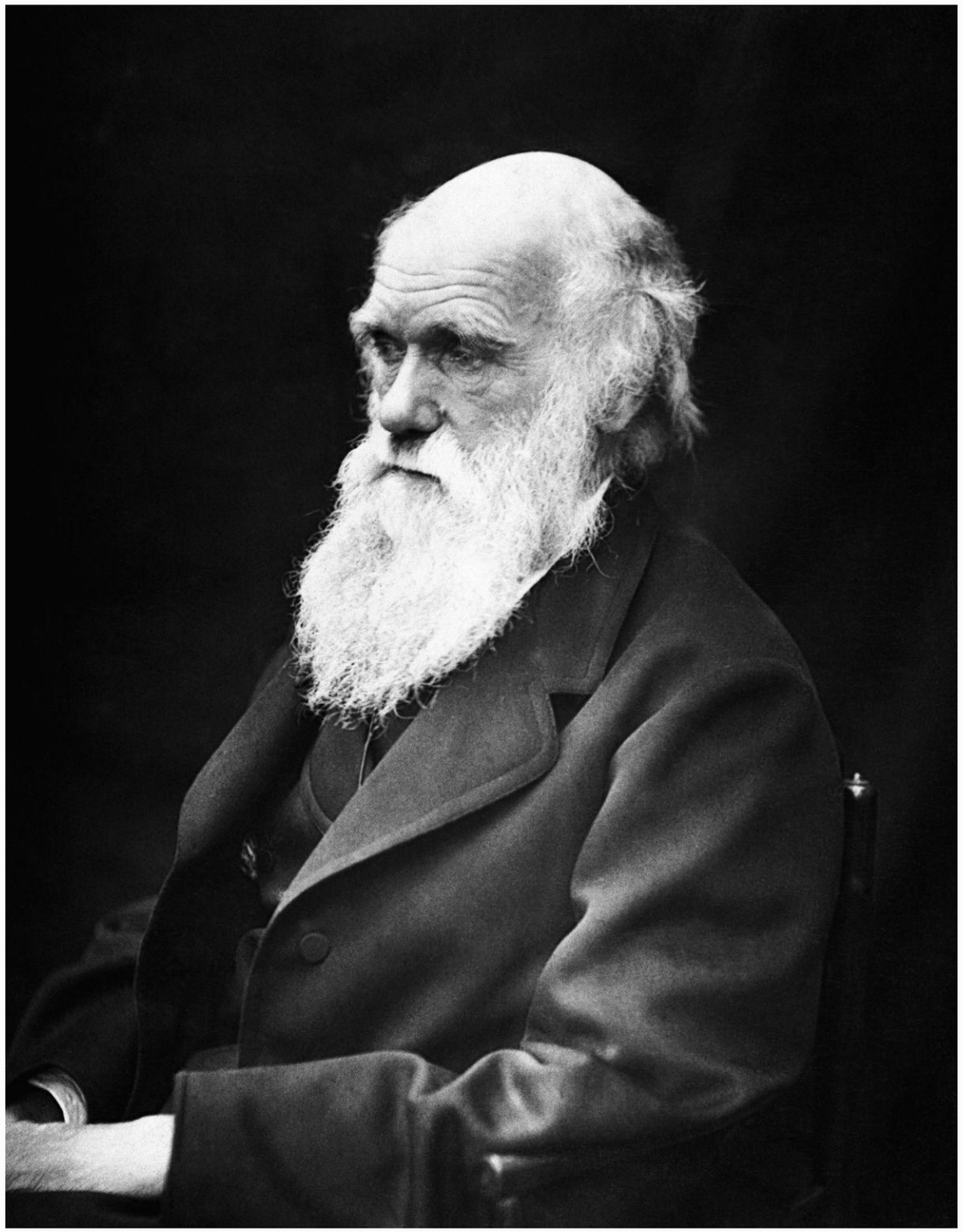
Algoritmi Genetici

Un po’ di contesto...



Algoritmi Genetici

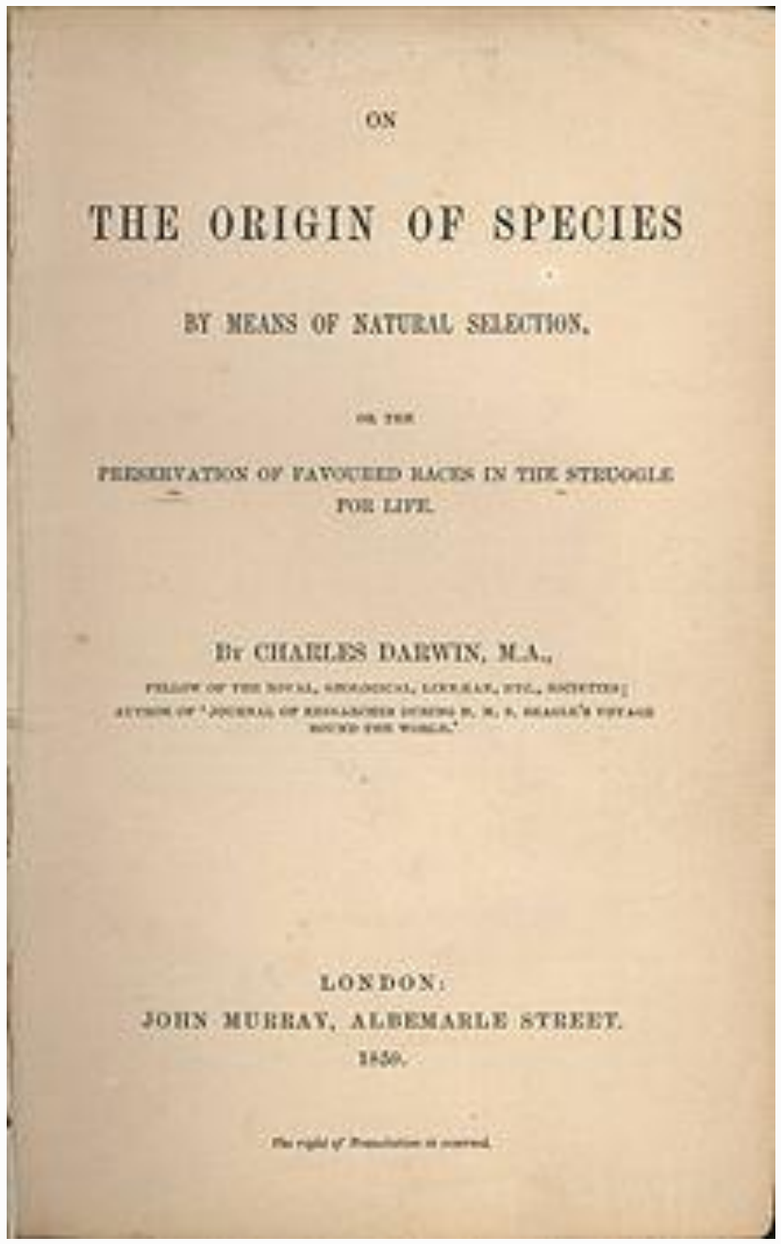
Un po’ di contesto...



Algoritmi Genetici

Un po' di contesto...

Il famoso scienziato Charles Darwin ne l'*Origine della Specie* (1859) ha introdotto i fondamenti della *teoria dell'evoluzione*. In essa, organismi della medesima specie **evolvono** tramite un processo che prende il nome di **selezione naturale**.


ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By CHARLES DARWIN, M.A.,

FELLOW OF THE ROYAL, GEOLOGICAL, LINNEAN, ETC., SOCIETIES;

AUTHOR OF 'JOURNAL OF RESEARCHES DURING H. M. S. BEAGLE'S VOYAGE ROUND THE WORLD.'

LONDON:

JOHN MURRAY, ALBEMARLE STREET.

1859.

The right of Translation is reserved.

Questa teoria è riassumibile in 4 punti cardine:

V *Variation*: gli individui in una popolazione differiscono nel patrimonio genetico (*genotipo*), che implica molte variazioni nelle loro caratteristiche fisiche (*fenotipo*).

I *Inheritance*: gli individui si riproducono e trasmettono parte del loro materiale genetico alla loro prole.

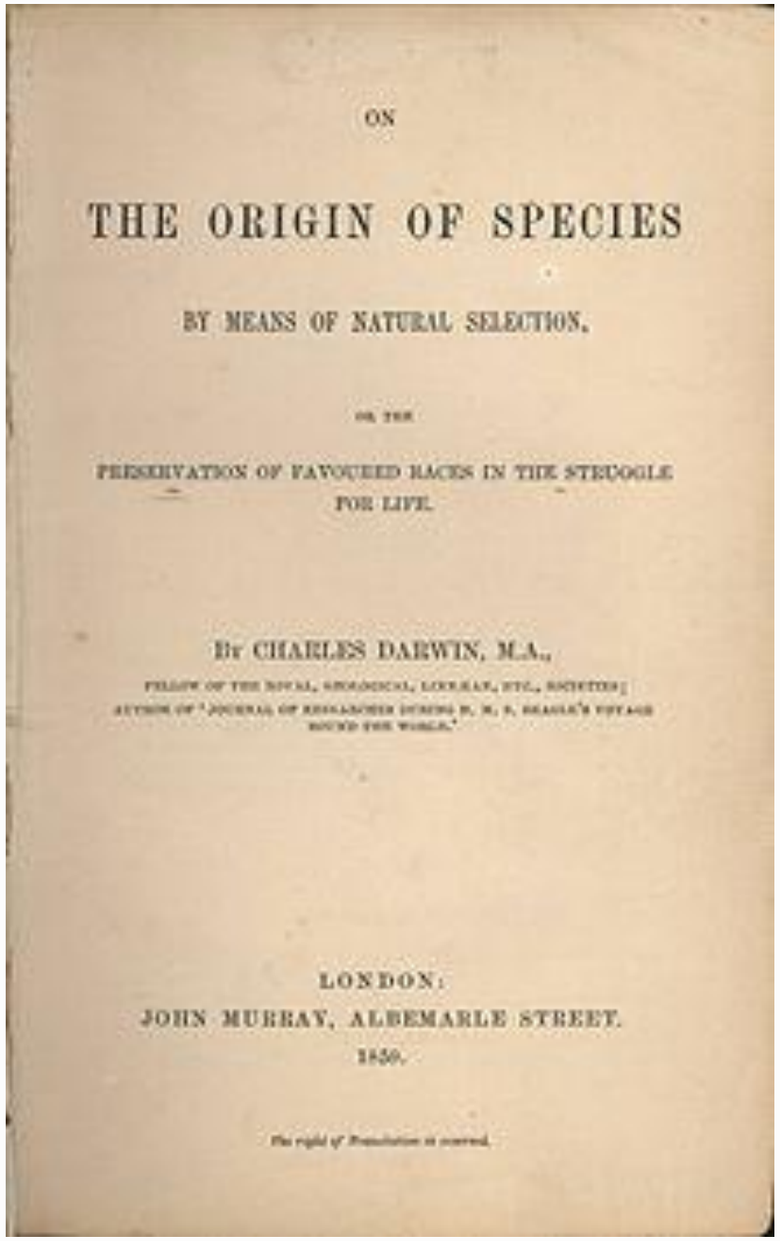
S *Selection (and Adaptation)*: alcuni individui possiedono tratti ereditati che gli permettono di (i) sopravvivere più a lungo in un *ambiente* e /o (ii) produrre ancora più prole. Di conseguenza, questi tratti tenderanno ad essere predominanti nell'intera popolazione.

T *Time*: a lungo andare (giorni, decenni, o anche milioni di anni), la selezione può comportare la nascita di nuove specie (*speciazione*).

Algoritmi Genetici

Un po' di contesto...

Il famoso scienziato Charles Darwin ne l'*Origine della Specie* (1859) ha introdotto i fondamenti della *teoria dell'evoluzione*. In essa, organismi della medesima specie **evolvono** tramite un processo che prende il nome di **selezione naturale**.


ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By CHARLES DARWIN, M.A.,

FELLOW OF THE ROYAL, GEOLOGICAL, LINNEAN, ETC., SOCIETIES;
AUTHOR OF 'JOURNAL OF RESEARCHES DURING H. M. S. BEAGLE'S VOYAGE ROUND THE WORLD.'

LONDON:
JOHN MURRAY, ALBEMARLE STREET.
1859.

The right of Translation is reserved.

Questa teoria è riassumibile in 4 punti cardine:

V *Variation*: gli individui in una popolazione differiscono nel patrimonio genetico (*genotipo*), che implica molte variazioni nelle loro caratteristiche fisiche (*fenotipo*).

I Probabilmente questo è il punto più importante della selezione naturale: gli individui più **adatti** permettono alla specie di sopravvivere.

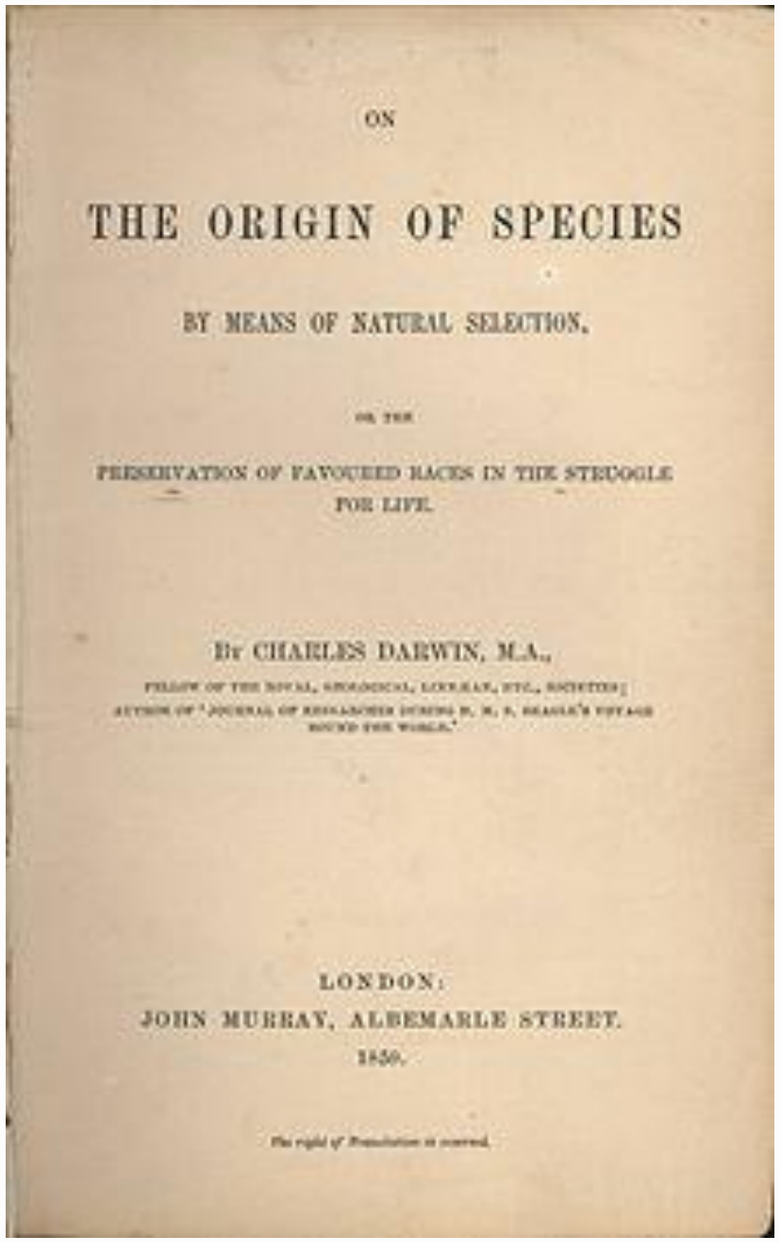
S *Selection (and Adaptation)*: alcuni individui possiedono tratti ereditati che gli permettono di (i) sopravvivere più a lungo in un *ambiente* e /o (ii) produrre ancora più prole. Di conseguenza, questi tratti tenderanno ad essere predominanti nell'intera popolazione.

T *Time*: a lungo andare (giorni, decenni, o anche milioni di anni), la selezione può comportare la nascita di nuove specie (*speciazione*).

Algoritmi Genetici

Un po' di contesto...

Il famoso scienziato Charles Darwin ne l'*Origine della Specie* (1859) ha introdotto i fondamenti della *teoria dell'evoluzione*. In essa, organismi della medesima specie **evolvono** tramite un processo che prende il nome di **selezione naturale**.


ON

THE ORIGIN OF SPECIES

BY MEANS OF NATURAL SELECTION,

OR THE

PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By CHARLES DARWIN, M.A.,

FELLOW OF THE ROYAL, GEOLOGICAL, LINNEAN, ETC., SOCIETIES;
AUTHOR OF 'JOURNAL OF RESEARCHES DURING H. M. S. BEAGLE'S VOYAGE ROUND THE WORLD.'

LONDON:
JOHN MURRAY, ALBEMARLE STREET.
1859.

The right of Translation is reserved.

Questa teoria è riassumibile in 4 punti cardine:

V *Variation*: gli individui in una popolazione differiscono nel patrimonio genetico (*genotipo*), che implica molte variazioni nelle loro caratteristiche fisiche (*fenotipo*).

I *Inheritance*: gli individui si riproducono e trasmettono parte del loro materiale genetico alla loro prole.

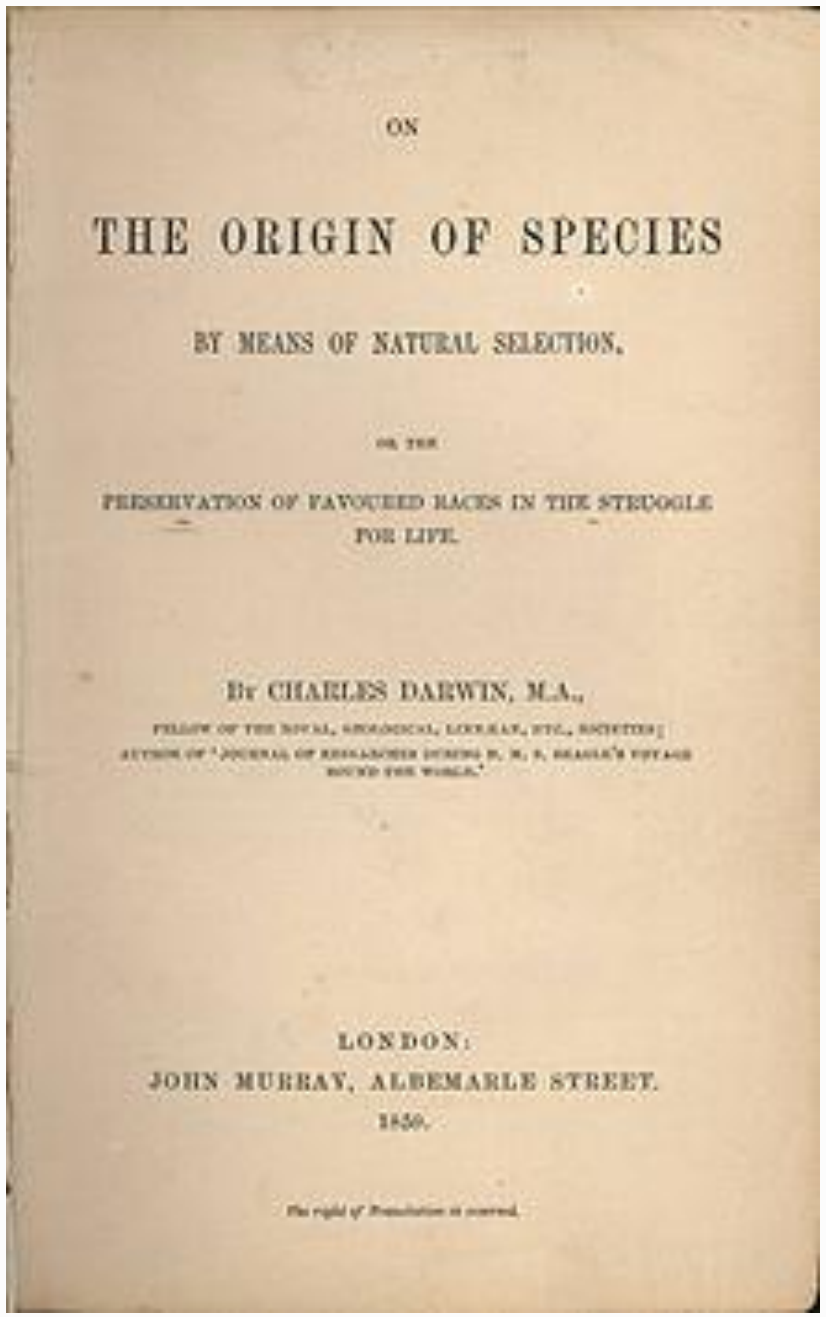
S [illegible] edono [illegible] re più [illegible] iù prole. Di conseguenza, questi tratti tenderanno ad essere predominanti nell'intera popolazione.

Anche questo aspetto è importante: gli individui si riproducono attraverso **generazioni**, trasmettendosi sia tratti positivi che negativi.

T *Time*: a lungo andare (giorni, decenni, o anche milioni di anni), la selezione può comportare la nascita di nuove specie (*speciazione*).

Un po' di contesto...

Il famoso scienziato Charles Darwin ne l'*Origine della Specie* (1859) ha introdotto i fondamenti della *teoria dell'evoluzione*. In essa, organismi della medesima specie **evolvono** tramite un processo che prende il nome di **selezione naturale**.



Questa teoria è riassumibile in 4 punti cardine:

- **V** *Variation*: gli individui in una popolazione differiscono nel patrimonio genetico (*genotipo*), che implica molte variazioni nelle loro caratteristiche fisiche (*fenotipo*).
- **I**

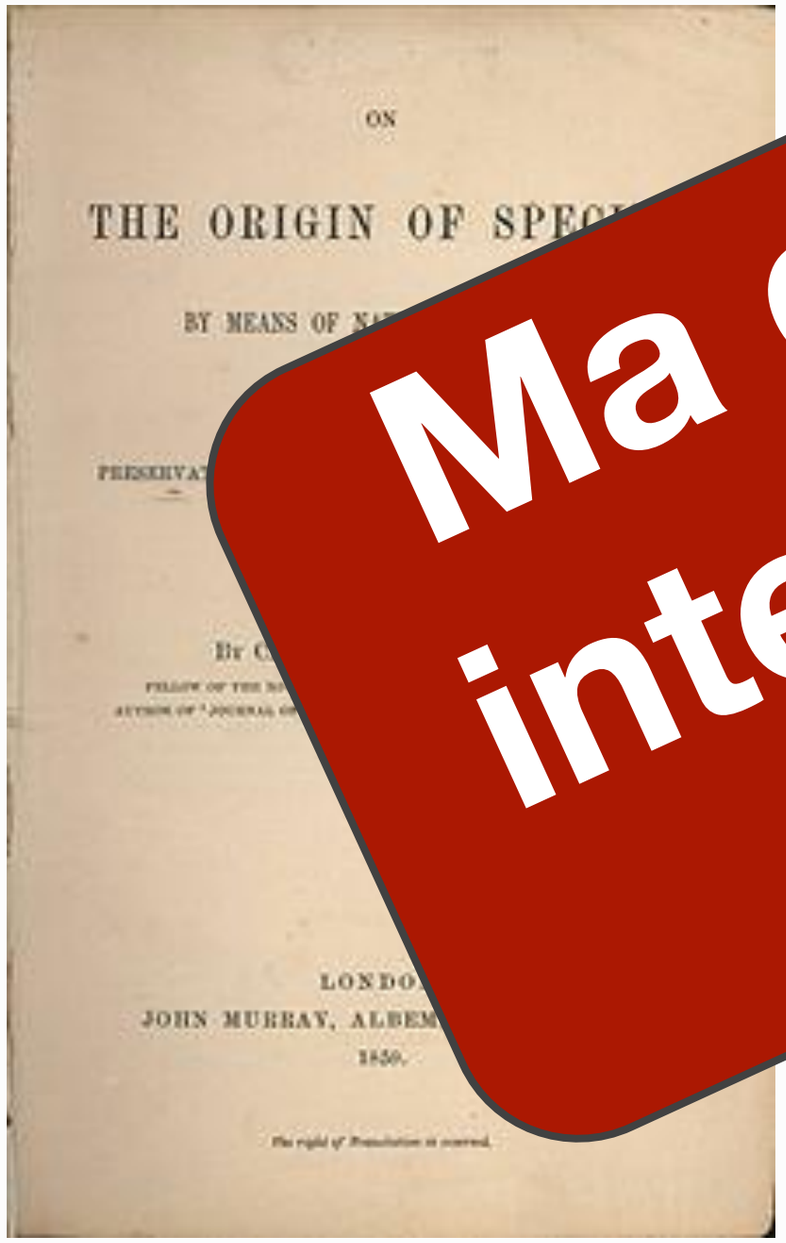
Non soltanto si riproducono, ma essi sono anche soggetti a **mutazioni** del loro corredo genetico.

- **S** *Selection (and Adaptation)*: alcuni individui possiedono tratti ereditati che gli permettono di (i) sopravvivere più a lungo in un *ambiente* e /o (ii) produrre ancora più prole. Di conseguenza, questi tratti tenderanno ad essere predominanti nell'intera popolazione.
- **T** *Time*: a lungo andare (giorni, decenni, o anche milioni di anni), la selezione può comportare la nascita di nuove specie (*speciazione*).

Algoritmi Genetici

Un po' di contesto...

Il famoso scienziato Charles Darwin ne l'*Origine della Spe* ntrodotto i fondamenti della *teoria dell'evoluzione*. In essa, orga a specie **evolvono** tramite un processo che prende il n



Questa t

eriscono
a molte
tipo).

ttono

ico.

Ma davvero ci interessa tutto ciò?

uni individui possiedono
mettono di (i) sopravvivere più
ente e /o (ii) produrre ancora più
seguenza, questi tratti tenderanno ad
predominanti nell'intera popolazione.

1. *Time*: a lungo andare (giorni, decenni, o anche milioni di anni), la selezione può comportare la nascita di nuove specie (*speciazione*).

Algoritmi Genetici

Teoria dell'Evoluzione e Ottimizzazione

Chiaramente, non siamo interessati alla genetica in sé. Ci interessa capire come riusare questi concetti per i nostri fini, soprattutto il concetto di **miglioramenti degli individui.**

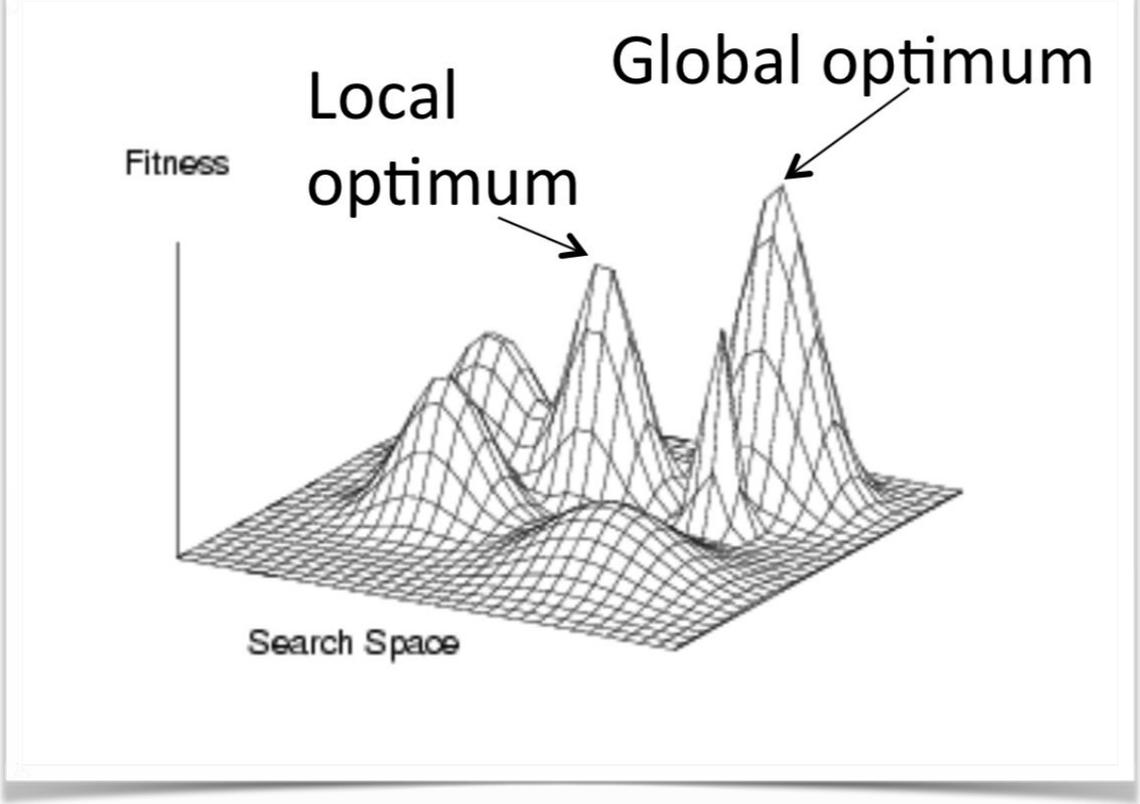
Chi sono questi "individui"?

Cosa si intende con "miglioramento"?

Migliorare rispetto a che cosa?

Supponiamo di avere una *funzione matematica* che vogliamo **massimizzare/minimizzare**, ovvero trovare la/le soluzione/i tali che il valore restituito dalla funzione sia il massimo/minimo **globale**. Un problema di questo tipo prende il nome di problema di **ottimizzazione**, e la funzione è nota come **funzione obiettivo.**

Esistono molti, moltissimi metodi ed algoritmi di ottimizzazione, che esplorano lo **spazio di ricerca** (a.k.a. regione ammissibile) con l'obiettivo di trovare il punto di ottimo globale.



Alcuni di questi sono **completi**, ovvero sicuramente in grado di trovare la soluzione (sotto certe condizioni e con sufficiente tempo di esecuzione concesso); mentre gli altri sono detti **incompleti**.

Algoritmi Genetici

Teoria dell'Evoluzione e Ottimizzazione

Chiaramente, non siamo interessati alla genetica in sé. Ci interessa capire come riusare questi concetti per i nostri fini, soprattutto il concetto di **miglioramenti degli individui.**

Chi sono questi "individui"?

Cosa si intende con "miglioramento"?

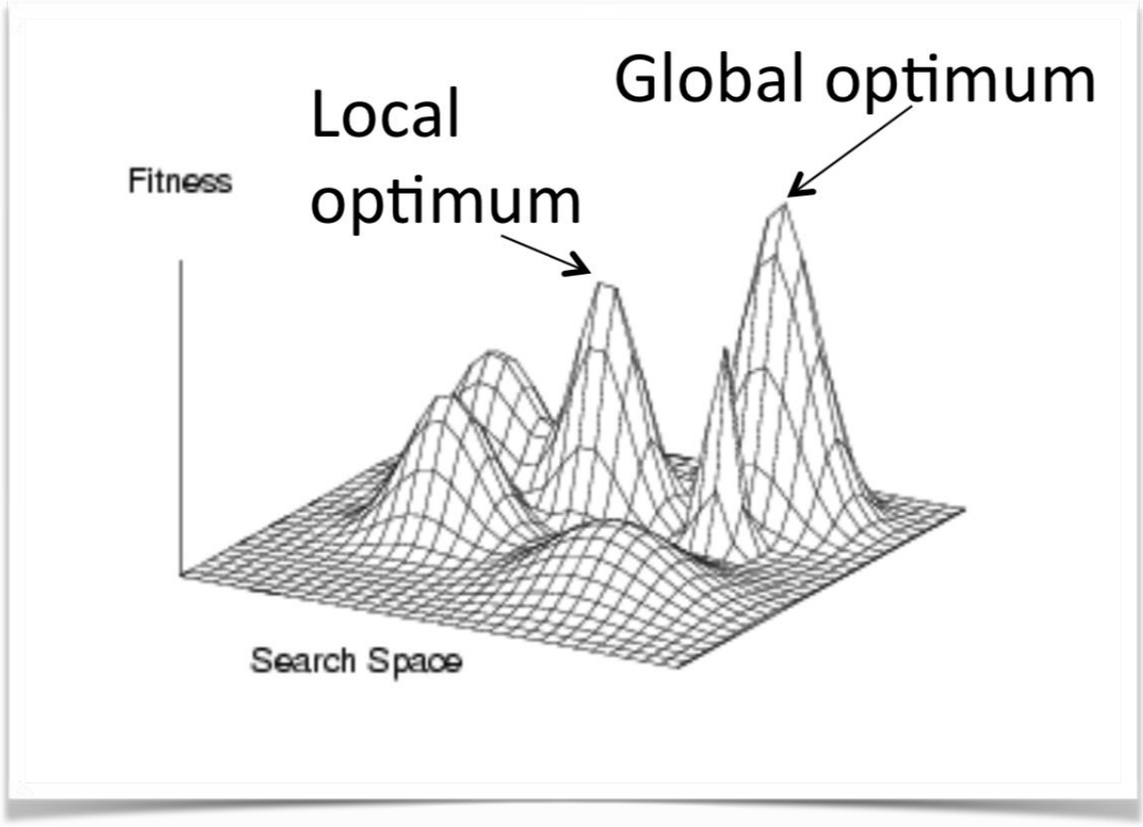
Migliorare rispetto a che cosa?

"

L'ottimo è nemico del buono

- *Qualcuno*

I metodi/algoritmi di ottimizzazione completi, con input sufficientemente complessi, non sono in grado di trovare la soluzione in tempi accettabili. E' necessario, perciò, accontentarsi di **soluzioni approssimate**, vicine all'ottimo globale, ma comunque accettabili per l'uso che ne si intende fare.



Inoltre, perché limitarsi a partire da una singola soluzione migliorandola passo passo (ricerca locale), quando potremmo "provare" diverse **soluzioni candidate** tutte insieme? Magari, potremmo *premiare* quelle più *promettenti* (vicine all'ottimo), e penalizzare quelle meno promettenti (più lontane). Questa idea è alla base della **ricerca globale**.

Algoritmi Genetici

Teoria dell'Evoluzione e Ottimizzazione

Chiaramente, non siamo interessati alla genetica in sé. Ci interessa capire come riusare questi concetti per i nostri fini, soprattutto il concetto di **miglioramenti degli individui.**

Chi sono questi "individui"?

Cosa si intende con "miglioramento"?

Migliorare rispetto a che cosa?

Adesso, proviamo a mappare tutti questi concetti con la teoria dell'evoluzione.

Questo mapping dà luogo ai cosiddetti **algoritmi evolutivi** (*Evolutionary Algorithms*, EA), ovvero algoritmi di ottimizzazione ispirati ai meccanismi della teoria dell'evoluzione.

Algoritmi Genetici

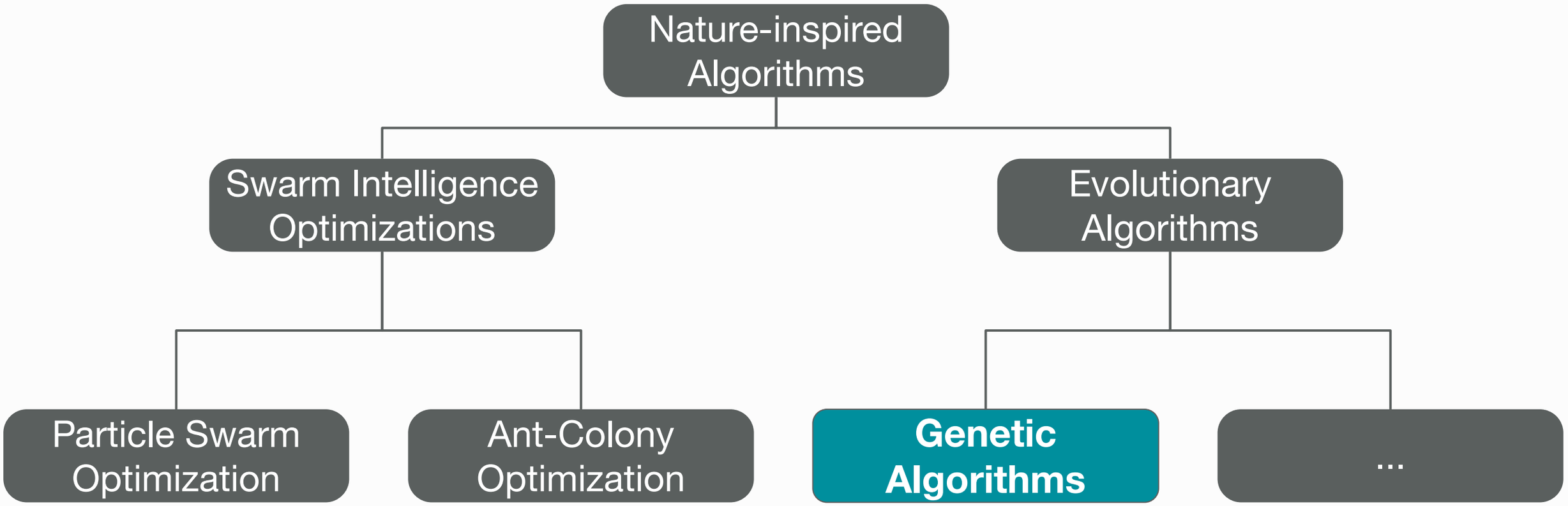
Teoria dell'Evoluzione e Ottimizzazione

Chiaramente, non siamo interessati alla genetica in sé. Ci interessa capire come riusare questi concetti per i nostri fini, soprattutto il concetto di **miglioramenti degli individui.**

Chi sono questi "individui"?

Cosa si intende con "miglioramento"?

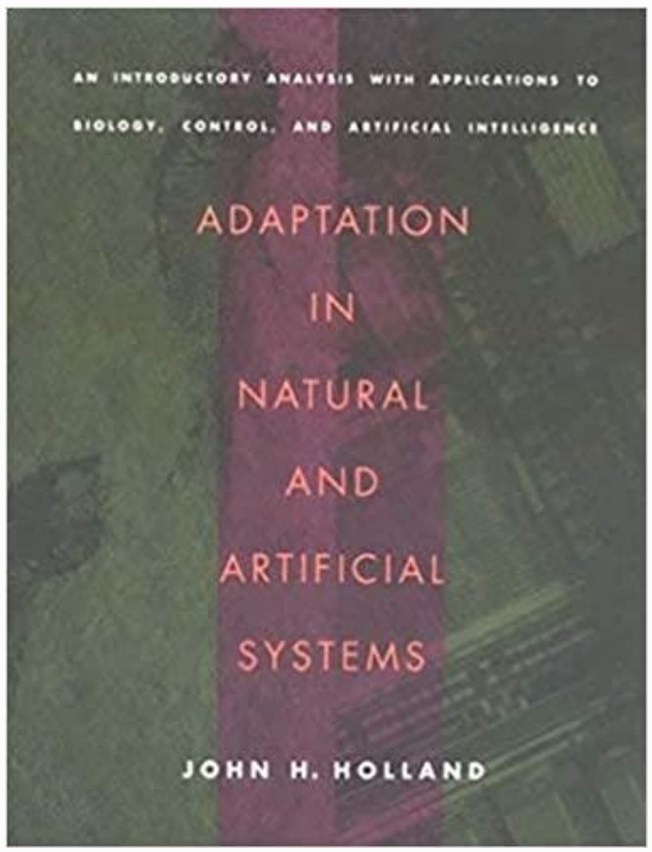
Migliorare rispetto a che cosa?



Gli algoritmi evolutivi sono un sotto-tipo degli *algoritmi ispirati dai meccanismi della natura*. Sicuramente gli algoritmi genetici (GA) sono il tipo più noto ed utilizzato di EAs, ma ne esistono altri (fuori dallo scope di questa lezione).

Algoritmi Genetici

Panoramica sui GAs



John Holland introduce il concetto di *genetic plans* nel libro *Adaptation in Natural and Artificial Systems, nel* 1975, più tardi rinominati in algoritmi genetici, come li conosciamo oggi. Sebbene siano stati presentati con un **forte fondamento teorico** (che tralasciamo), oggi i GAs sono usati per la loro estrema praticità e flessibilità.

> ***Algoritmo Genetico:*** *procedura ad alto livello (metaeuristica) ispirata dalla genetica per definire un algoritmo di ottimizzazione capace di esplorare in maniera efficiente lo spazio di ricerca.*

GAs, come le altre metaeuristiche, non sono algoritmi già definiti e pronti all'uso. Possiamo vederli come dei **framework per definire un nostro algoritmo di ottimizzazione**. In quanto meta-euristica essi sono:

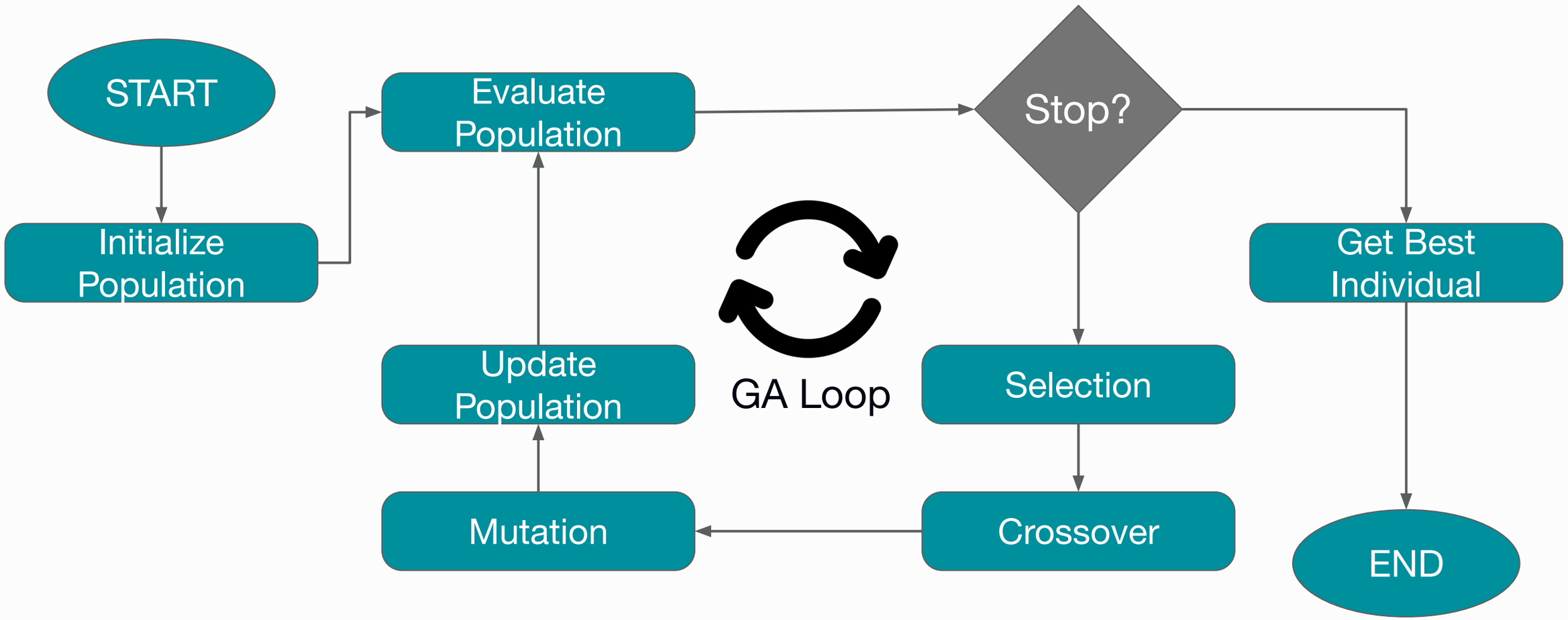
- **Slegati dallo specifico problema**: sono in grado di risolvere un'ampia gamma di problemi di ottimizzazione e di ricerca;
- **Efficienti**: esplorano in maniera rapida gran parte dello spazio di ricerca;
- **Approssimati**: forniscono soluzioni sub-ottimali.

Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*

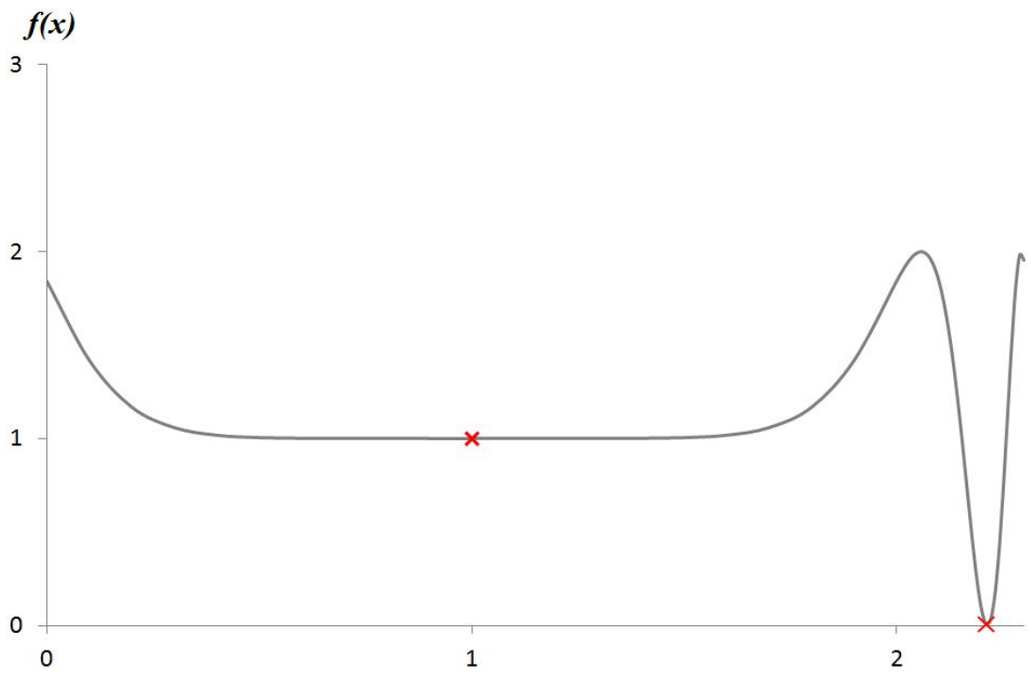


Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*

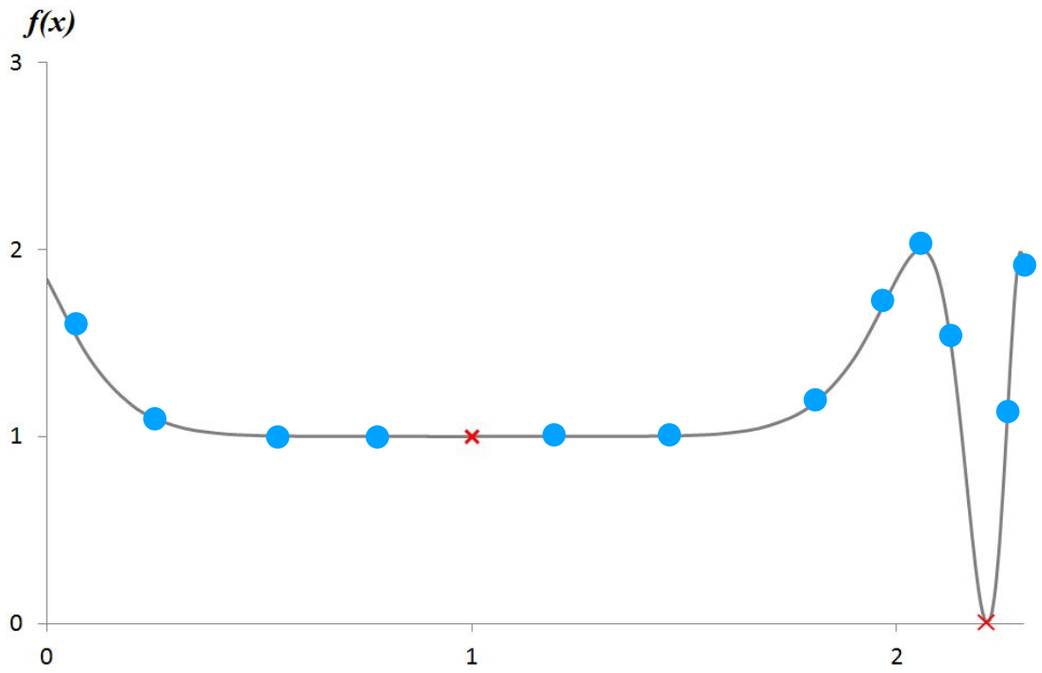


Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



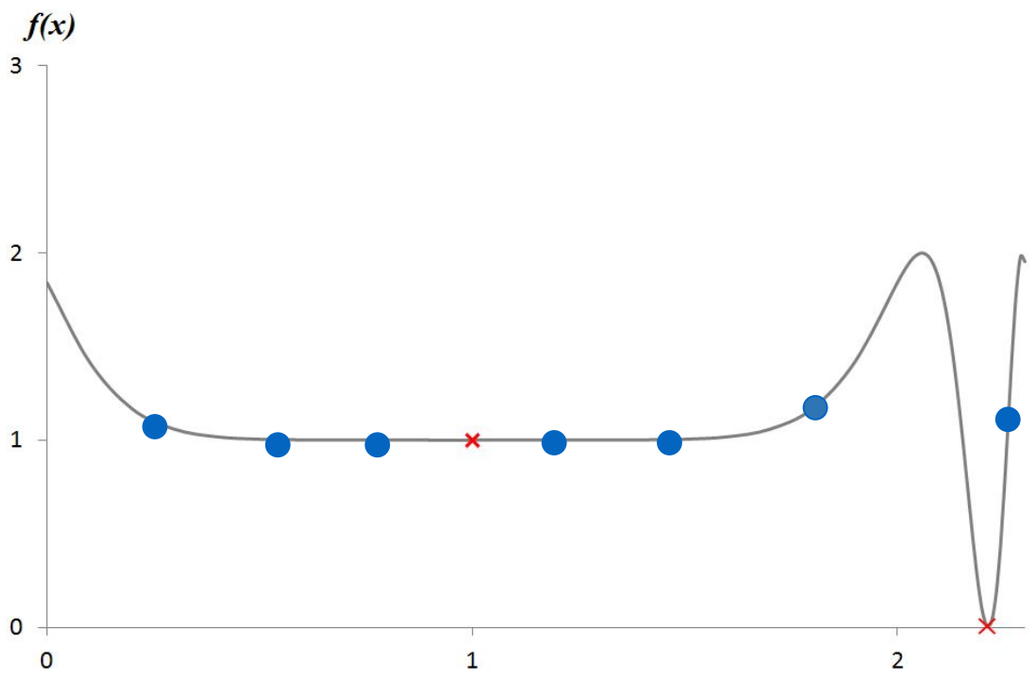
Initial Population

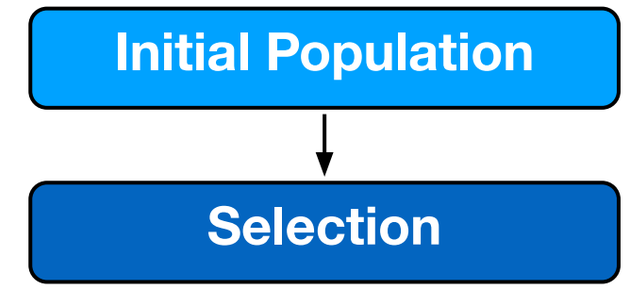
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



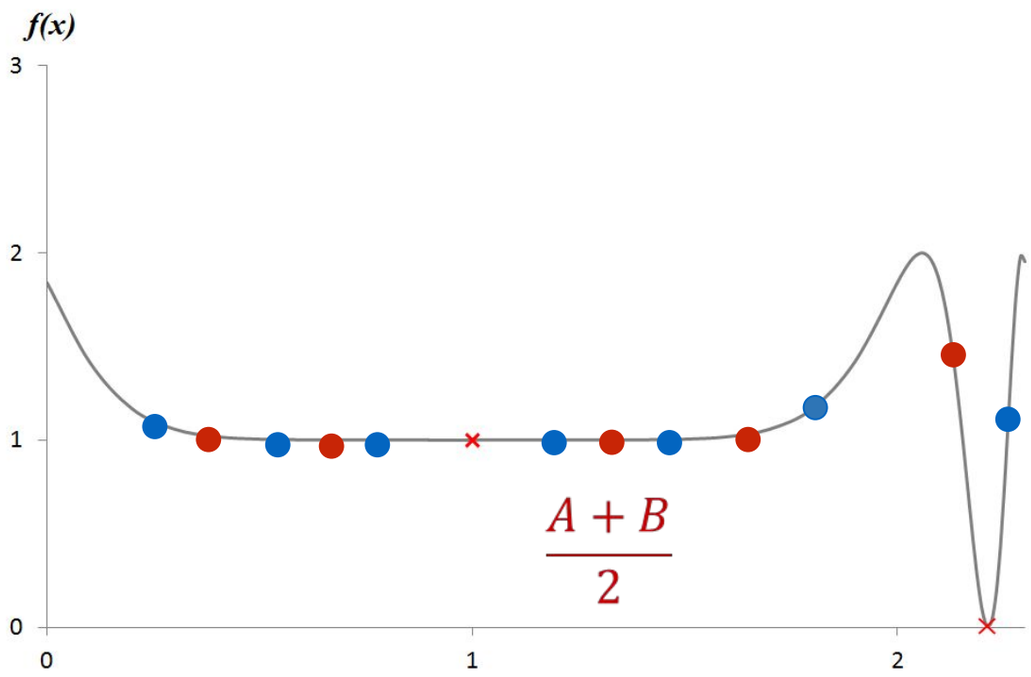


Algoritmi Genetici

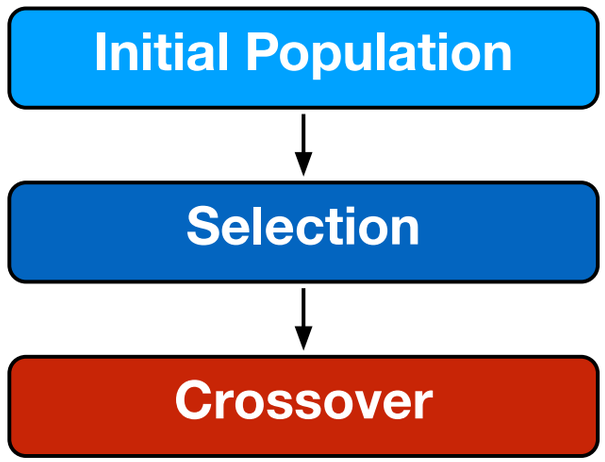
Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



- Current population
- New solutions (offsprings)

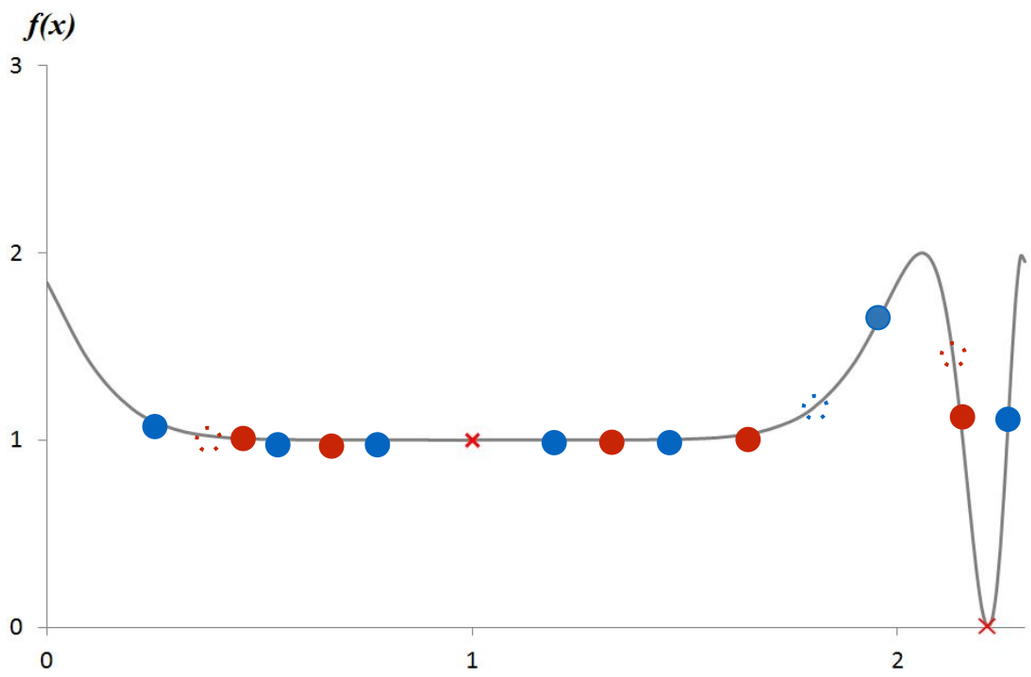


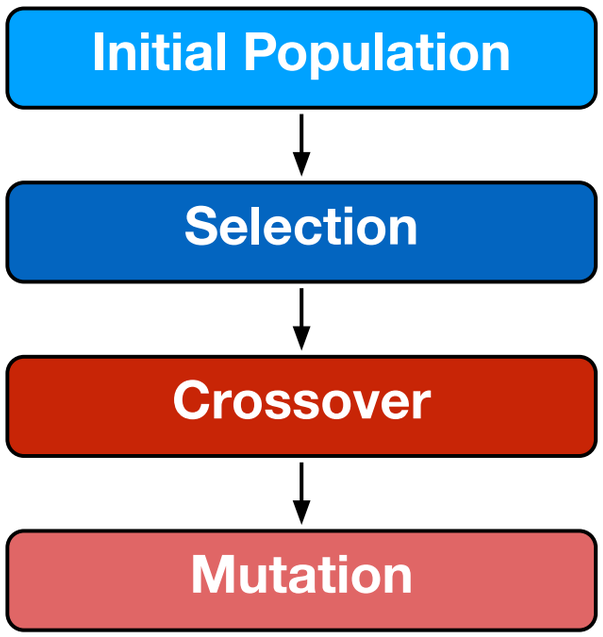
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*





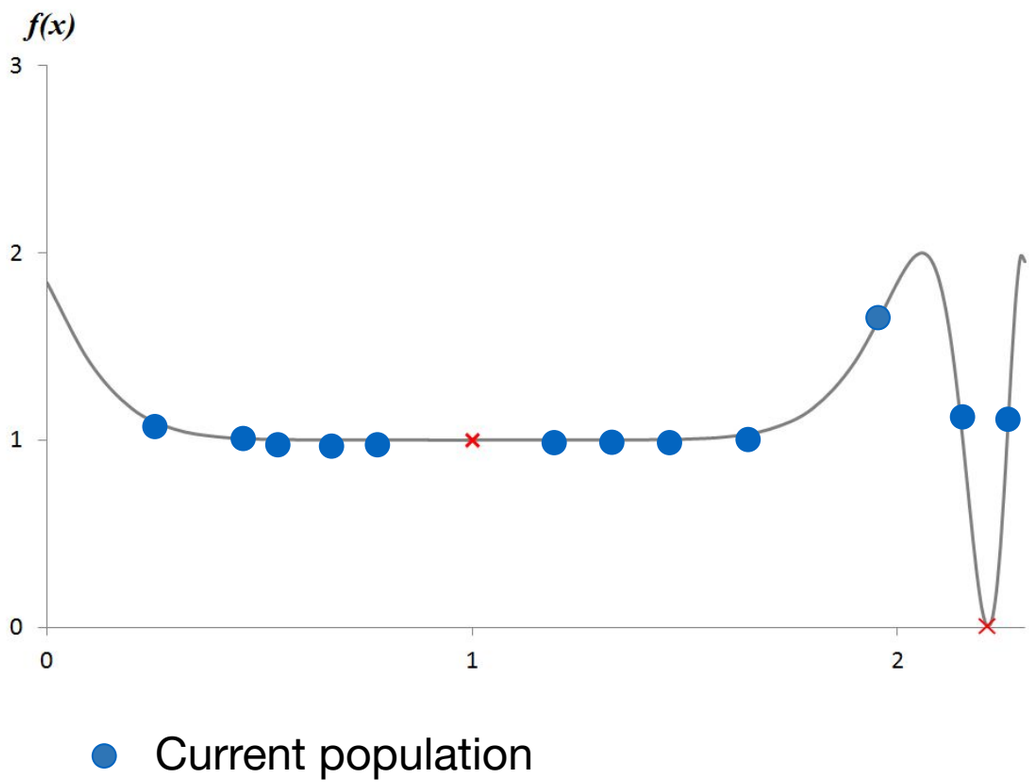
- Current population
- New solutions (offsprings)

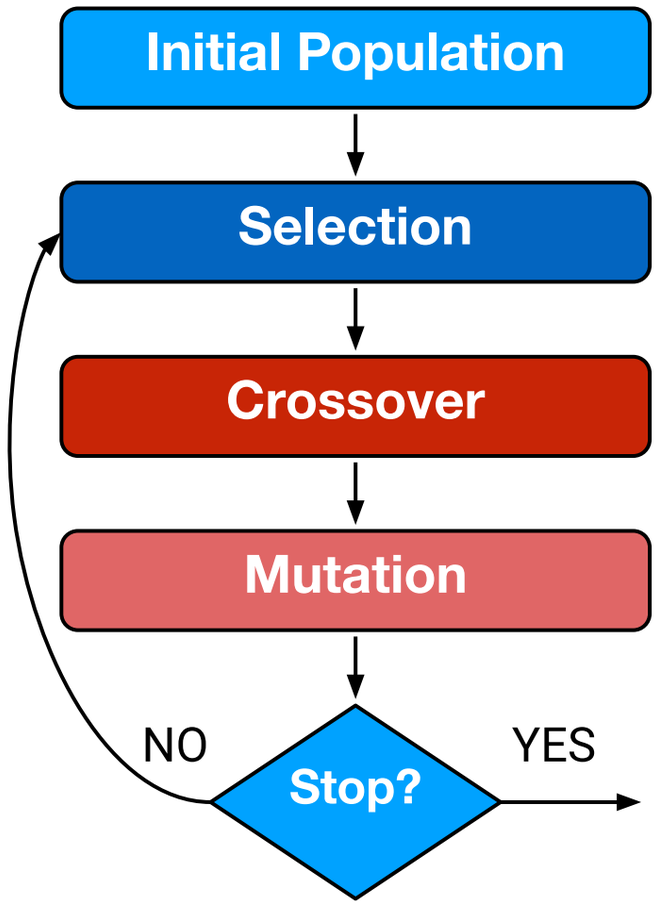
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



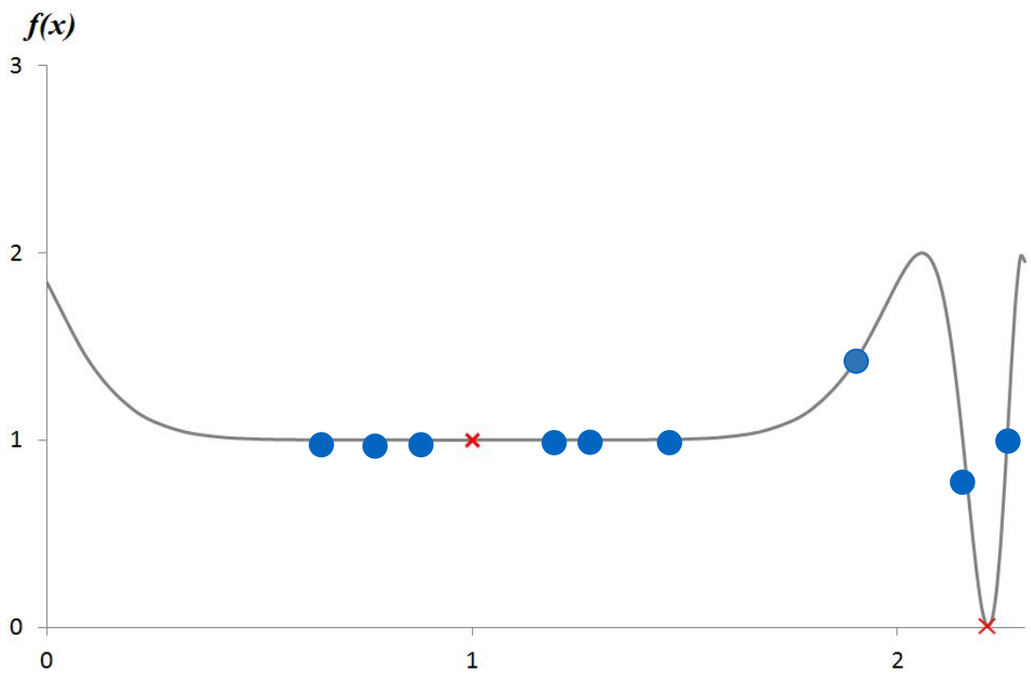


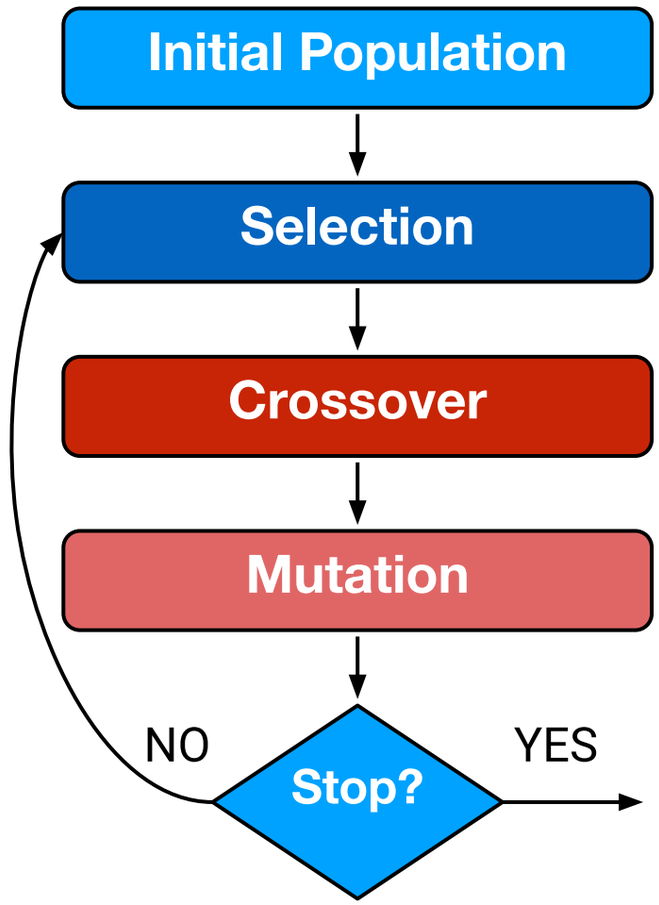
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



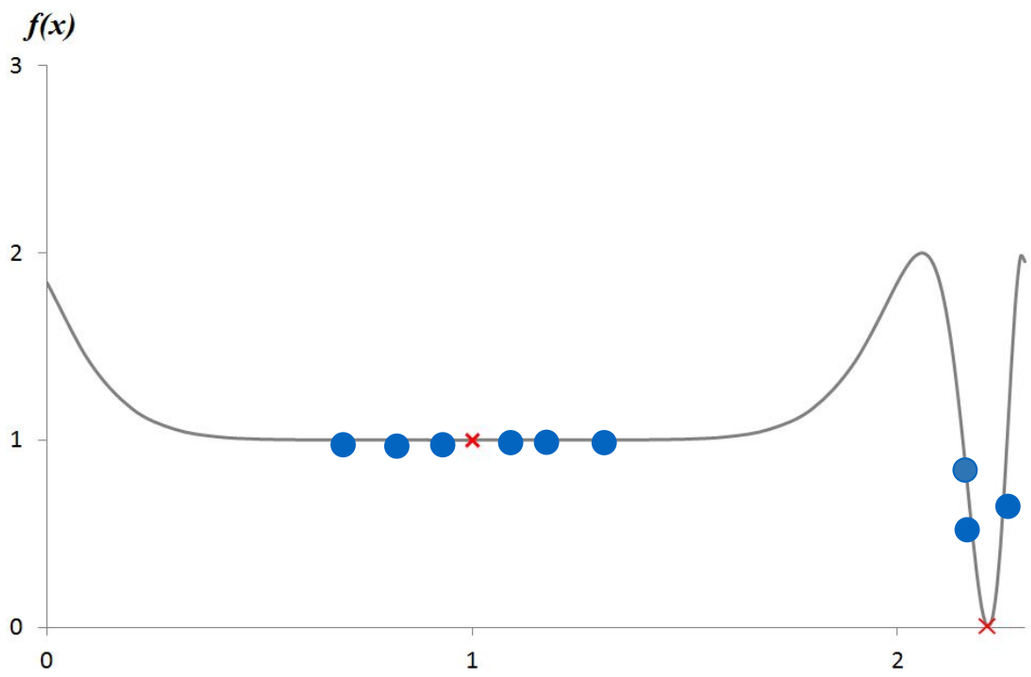


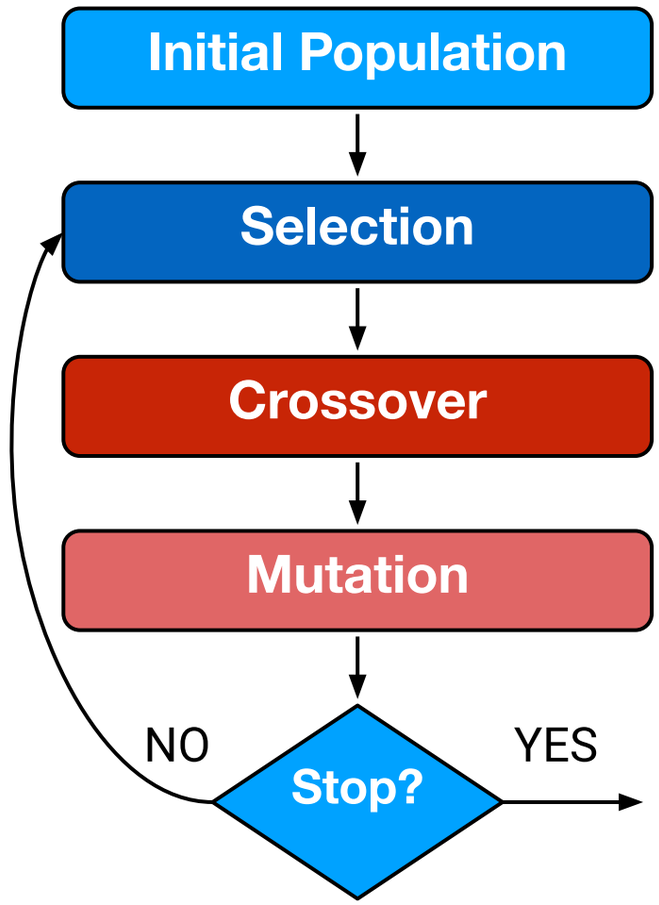
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*



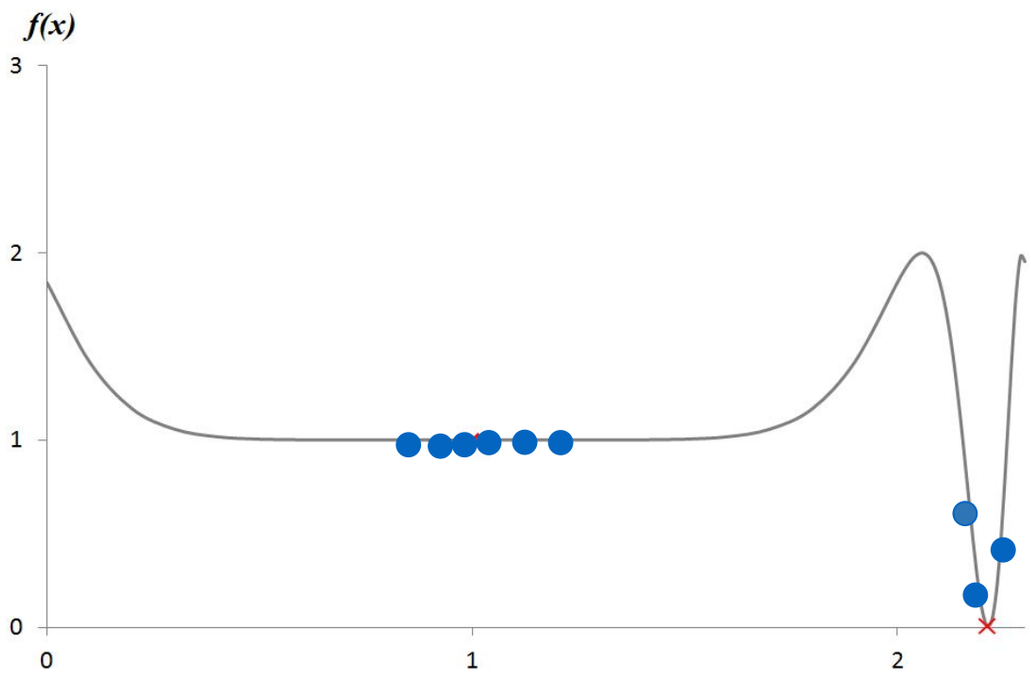


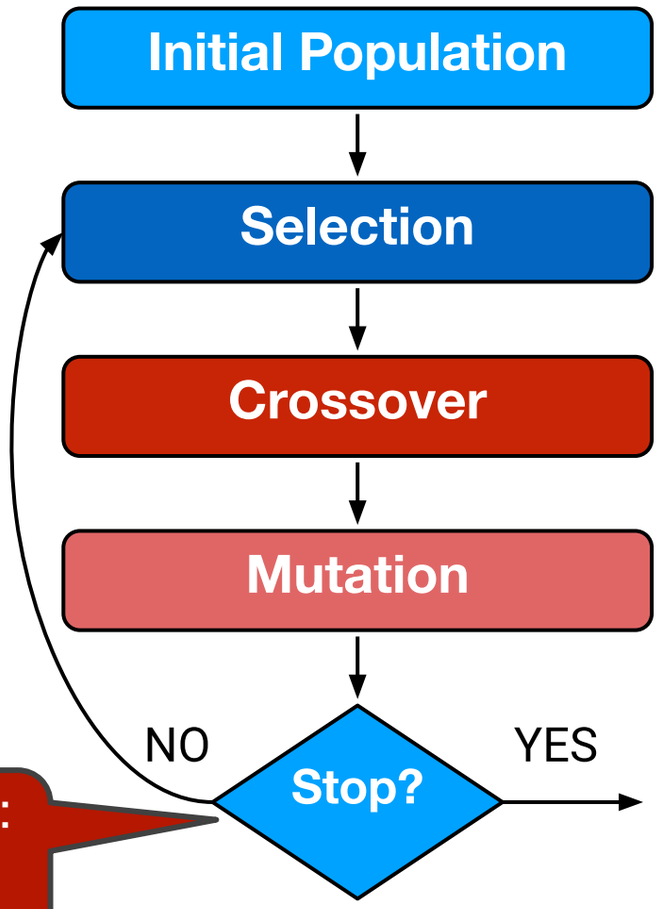
Algoritmi Genetici

Panoramica sui GAs

Proviamo a sintetizzare cosa fa un algoritmo genetico in buona sostanza:

*Un GA fa evolvere **iterativamente** una **popolazione** di **individui** (soluzioni candidate), producendo di volta in volta nuove **generazioni** di individui migliorati, rispetto ad una cosiddetta **misura di fitness**, finché uno o più **criteri di arresto** non sono soddisfatti. La generazioni di nuovi individui avviene per mezzo di **tre operatori di ricerca**, quali **selezione**, **crossover** e **mutazione**.*





Un tipico criterio di arresto è stabilito dal **tempo di esecuzione**: l'evoluzione termina dopo dopo X secondi di esecuzione. Questo X prende il nome di **budget di ricerca.**

Algoritmi Genetici

Panoramica sui GAs

Goldberg identifica 4 **caratteristiche fondamentali** dei GAs:

Encoding

Un GA necessita di un'appropriata **rappresentazione codificata degli individui.** Comunemente (ma non necessariamente) si adoperano stringhe binarie di lunghezza finita. Questo è probabilmente l'aspetto che determina la fattibilità di una soluzione basata su algoritmi genetici.

Global Search

Un GA usa un insieme di soluzioni candidate per esplorare da più punti lo spazio di ricerca. Questo riduce il rischio di "bloccarsi in ottimi locali".

Blindness

Un GA non ha bisogno di conoscere internamente la funzione da ottimizzare, né ha bisogno di particolari assunzioni. E' necessario che sia soltanto **accessibile** in maniera black-box. Qualora non dovesse esserlo, si possono adottare delle approssimazioni.

Probabilistic

Un GA include **componenti di casualità** che servono a guidare la ricerca (senza degenerare in un algoritmo *random search*).

Algoritmi Genetici

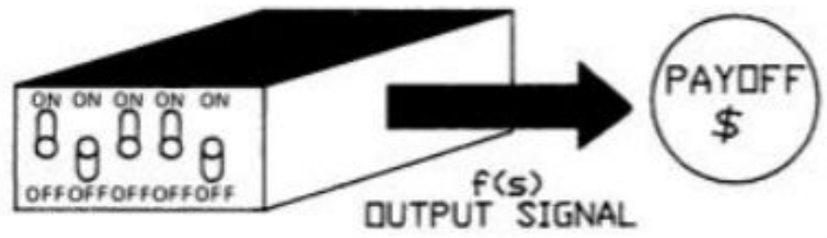
Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

Un GA si presta molto bene:

- ci basta che la funzione obiettivo sia accessibile (**blindness**) e poco complessa da calcolare;
- una soluzione candidata potrebbe essere **codificata** come una stringa lunga 5 bit.



La configurazione in figura si potrebbe codificare con la stringa 10110.

Cosa facciamo a stabilire se un individuo sia “migliore” di un altro? Abbiamo bisogno di una **funzione di valutazione**, ovvero una funzione che prende in input la codifica di un individuo e restituisce un valore che rappresenta la bontà di tale individuo.

Quale scegliamo? Dato che la funzione obiettivo $f(s)$ è accessibile e non richiede molte risorse per essere calcolata, possiamo **riusarla come funzione di valutazione**. Questo riuso avviene nella maggior parte dei casi (tant’è che spesso i due concetti sono usati come sinonimi), ma in altri casi la funzione obiettivo potrebbe essere complessa da calcolare o addirittura impossibile. Quindi, conviene definire la funzione di valutazione come un’**approssimazione** della vera funzione obiettivo.

Algoritmi Genetici

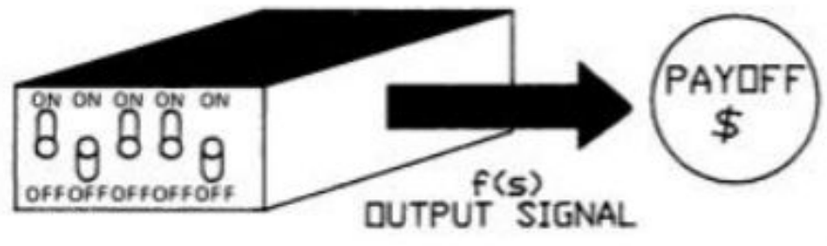
Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

Un GA si presta molto bene:

- ci basta che la funzione obiettivo sia accessibile (**blindness**) e poco complessa da calcolare;
- una soluzione candidata potrebbe essere **codificata** come una stringa lunga 5 bit.



La configurazione in figura si potrebbe codificare con la stringa 10110.

Cosa facciamo a stabilire se un individuo sia “migliore” di un altro? Abbiamo bisogno di una **funzione di valutazione**, ovvero una funzione che prende in input la codifica di un individuo e restituisce un valore che rappresenta la bontà di tale individuo.

Quale scegliamo? Dato che la funzione obiettivo $f(s)$ è accessibile e non richiede molte risorse per essere calcolata, possiamo **riusarla come funzione di valutazione**. Questo riu sso i due concetti sono usati co bbe essere complessa da calcolare o addirittura impossibile. Qu nviene definire la funzione di valutazione come un’**approssimazione** della vera funzione obiettivo.

E’ importantissimo che la funzione di valutazione sia veloce, visto che sarà necessario eseguirla un numero elevatissimo di volte ad ogni iterazione.

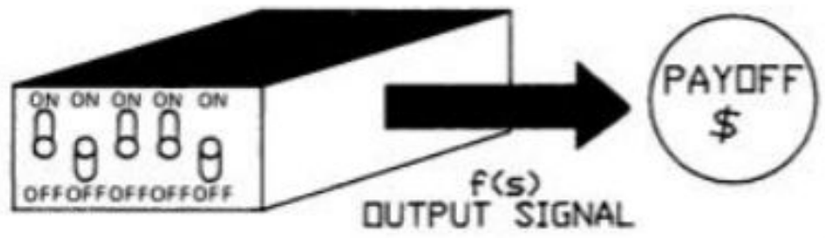
Algoritmi Genetici

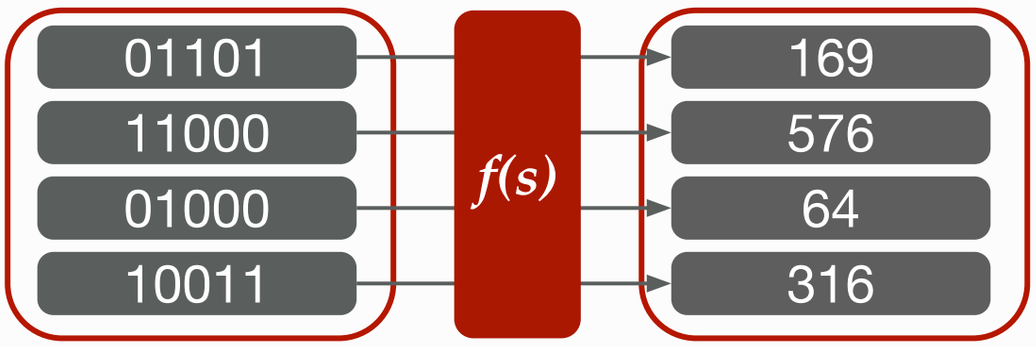
Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

Un GA iniziare sempre da una **popolazione iniziale** (prima generazione), che creaiamo **casualmente**. Supponiamo di fissare la taglia della popolazione a solo **4 individui**.





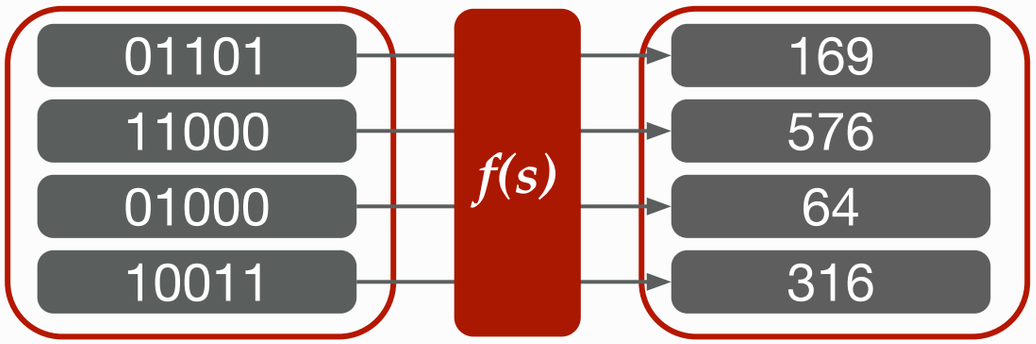
I valori ottenuti rappresentano il punto di partenza per il calcolo del grado di sopravvivenza di ciascun individuo. Non possiamo usare la funzione di valutazione, ma ci serve una **funzione di fitness** dedicata a tale scopo. La scelta della funzione di fitness è determinata dal **criterio di selezione** che decidiamo di usare.

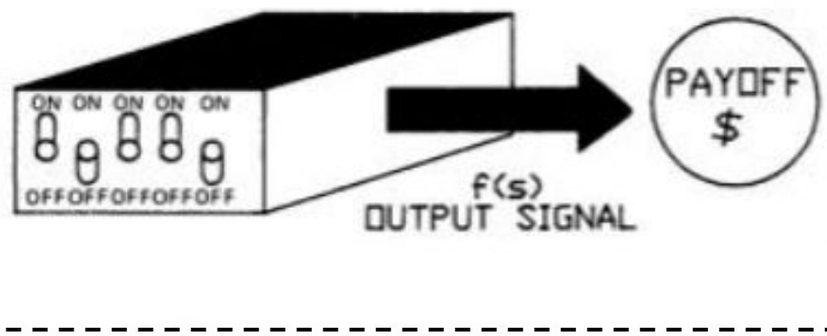
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1 **Selezione**

Un sottoinsieme della popolazione viene **candidato alla riproduzione** e duplicato in una generazione intermedia (in questa fase anche detta *mating pool*). Il criterio di selezione dovrebbe favorire gli individui con alta fitness (maggiore probabilità di essere selezionati).

2 **Crossover**

Gli individui selezionati (*parents*) sono **abbinati casualmente** per generare della prole (*offsprings*) incrociando il loro corredo genetico (la codifica). I figli generati **rimpiazzeranno** i genitori nella generazione intermedia. Il crossover è anche noto come *recombination.*

3 **Mutazione**

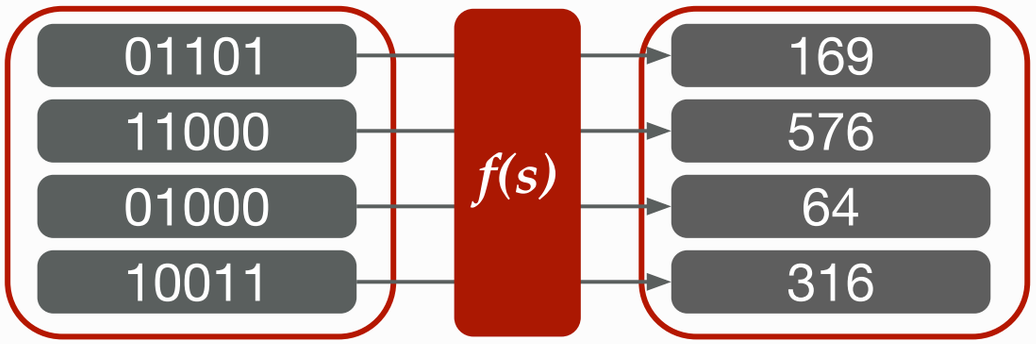
Alcuni *geni* (singoli elementi delle codifiche) presenti tra tutti gli individui sono **mutati casualmente** (bassa probabilità). La mutazione, di solito, non deve dar ad individui con codifiche fuori dal dominio del problema.

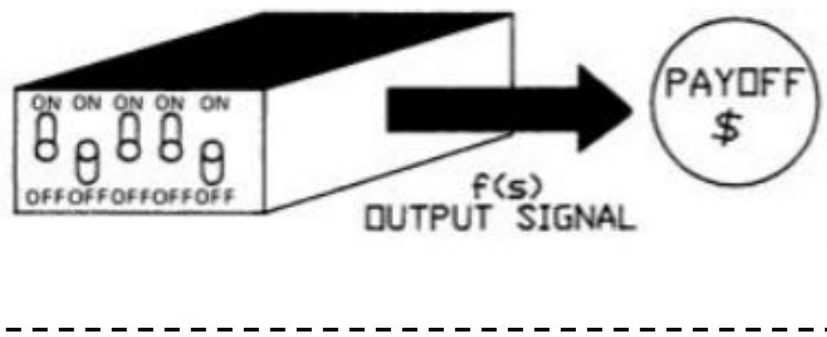
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1 Selezione

Un sottoinsieme della popolazione viene **candidato alla riproduzione** e duplicato in una generazione intermedia (in questa fase anche detta *mating pool*). Il criterio di selezione dovrebbe favorire gli individui con alta fitness (maggiore probabilità di essere selezionati).

La selezione permette di sfruttare (**exploit**) di buona soluzioni. Infatti, le soluzioni più promettenti sono favorite nella speranza che esse permettano la generazione di altre più promettenti.

Gli individui selezionati (*parents*) sono **abbinati casualmente** per generare oro corredo genetico (la codifica). I figli ella generazione intermedia. Il *bination.*

3 Mutazione

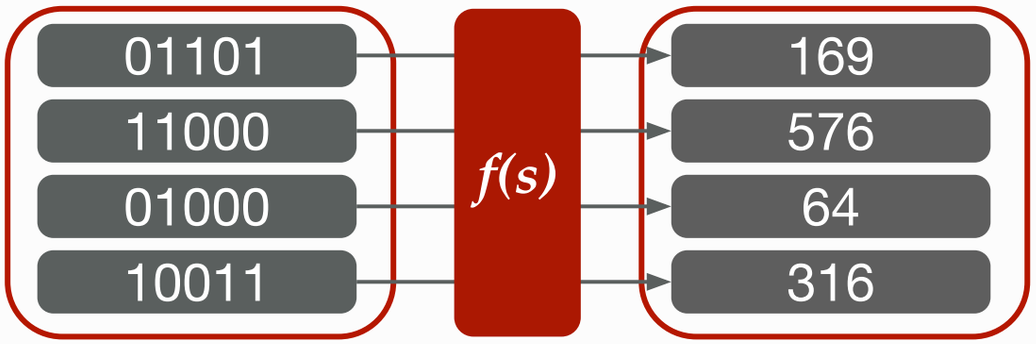
Alcuni *geni* (singoli elementi delle codifiche) presenti tra tutti gli individui sono **mutati casualmente** (bassa probabilità). La mutazione, di solito, non deve dar ad individui con codifiche fuori dal dominio del problema.

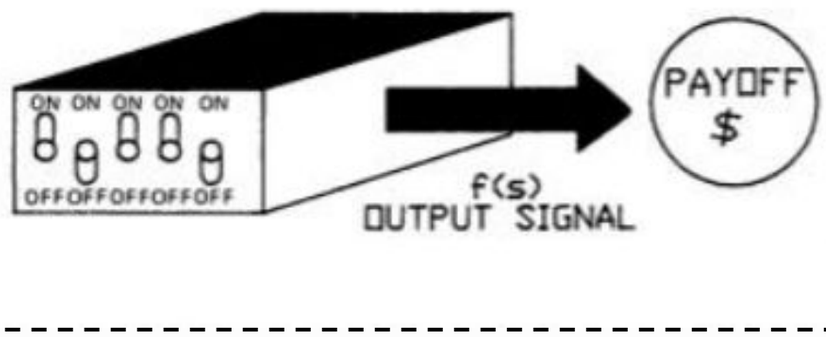
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





Un sottoinsieme della popolazione viene **candidato alla riproduzione** e
dia (in questa fase anche detta *mating*
favorire gli individui con alta fitness
onati).

Il crossover e la mutazione permettono di esplorare (**explore**) efficientemente lo spazio di ricerca, andando a ridurre il rischio di bloccarsi in ottimi locali.

2 Crossover

Gli individui selezionati (*parents*) sono **abbinati casualmente** per generare della prole (*offsprings*) incrociando il loro corredo genetico (la codifica). I figli generati **rimpiazzeranno** i genitori nella generazione intermedia. Il crossover è anche noto come *recombination.*

3 Mutazione

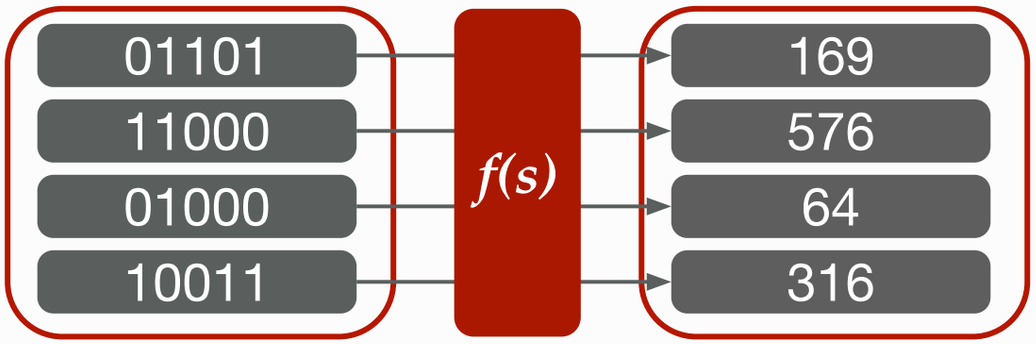
Alcuni *geni* (singoli elementi delle codifiche) presenti tra tutti gli individui sono **mutati casualmente** (bassa probabilità). La mutazione, di solito, non deve dar ad individui con codifiche fuori dal dominio del problema.

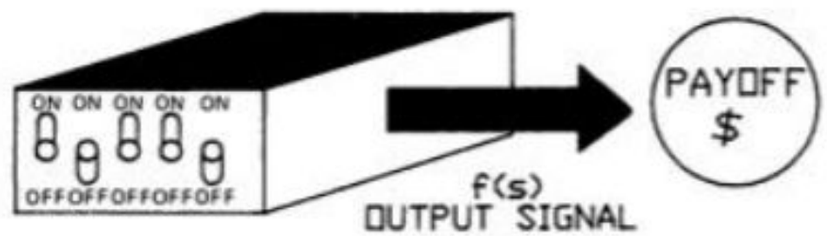
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





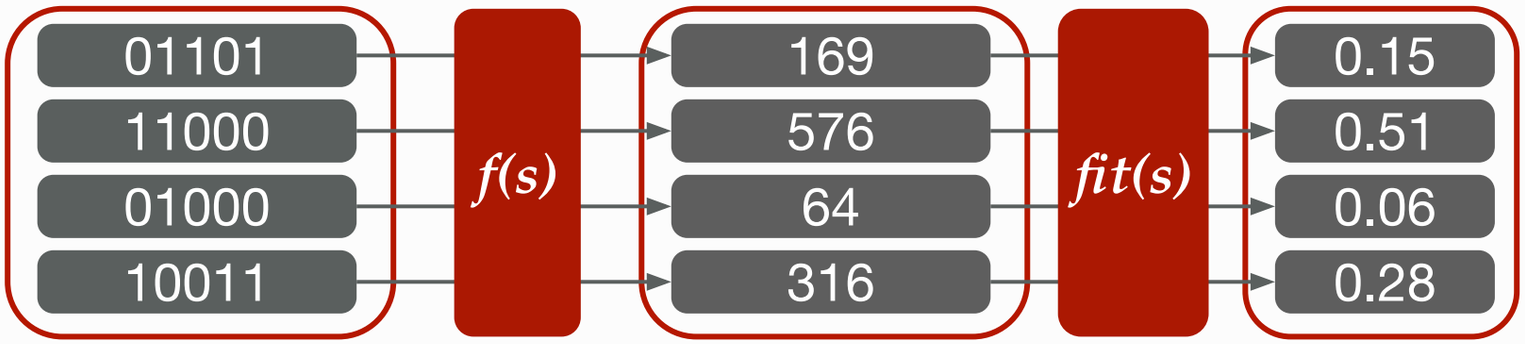
1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

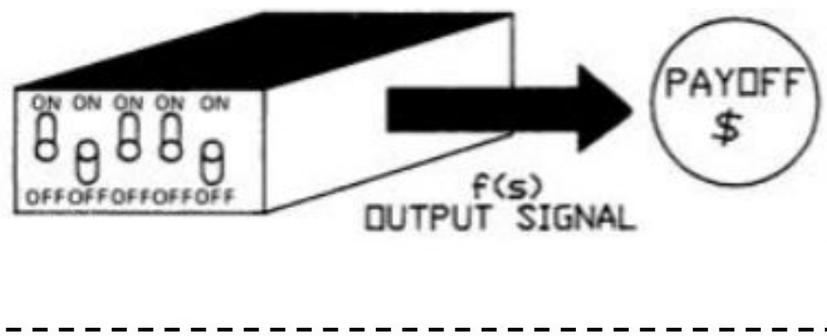
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1. **Roulette Wheel Selection**

Ogni individuo riceve una porzione di una ruota proporzionata al valore di valutazione relativo al resto delle valutazioni. In questo caso, la funzione di fitness è un valore compreso tra 0 e 1:

$$fit(s) = f(s) \,/\, \Sigma\, f(s)$$

2. **Single-point Crossover**

La ruota viene girata un numero di volte pari al numero degli individui (4), e i vincitori sono ammessi al mating pool. Ne segue che un individuo può vincere più volte, quindi potrebbero esserci dei cloni nel mating pool.

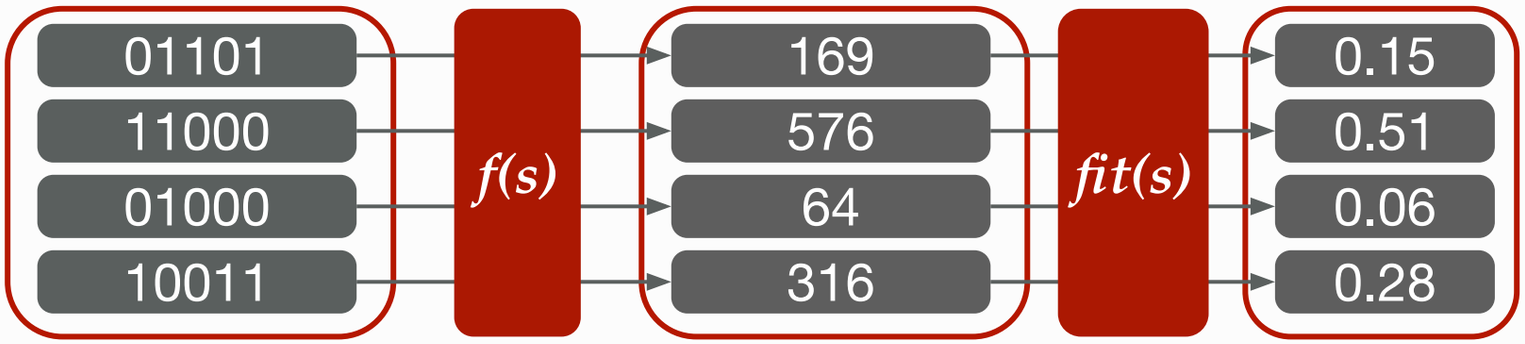
3. **Bit-Flip Mutation**

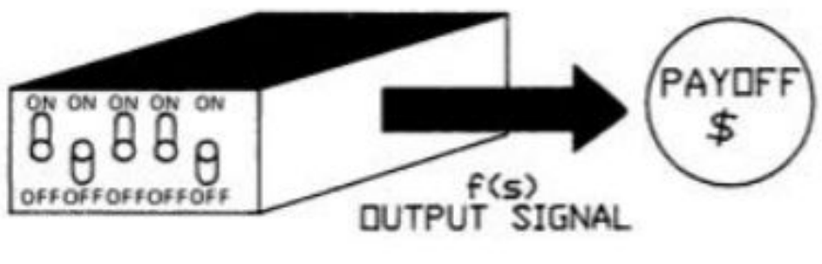
Algoritmi Genetici

Running Example

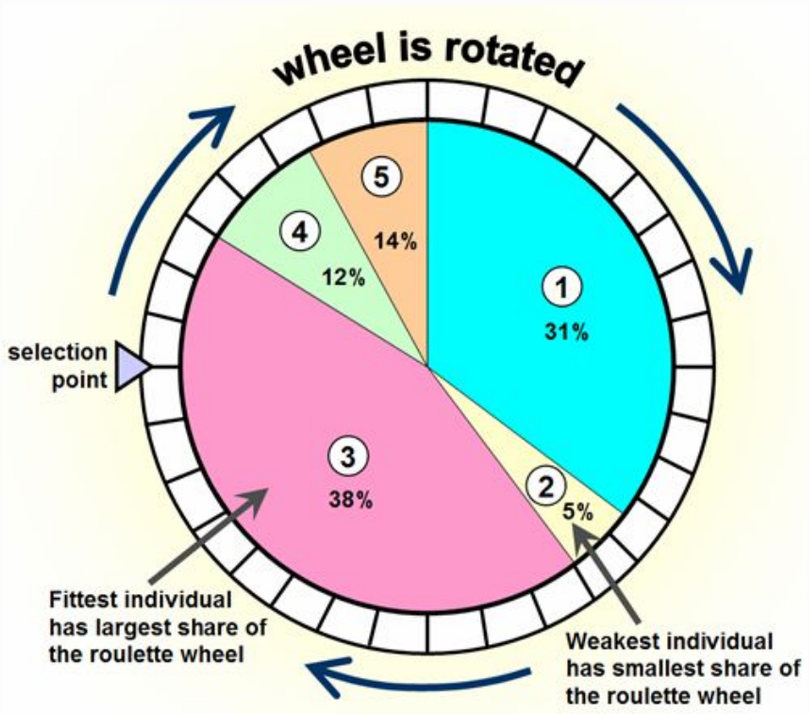
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

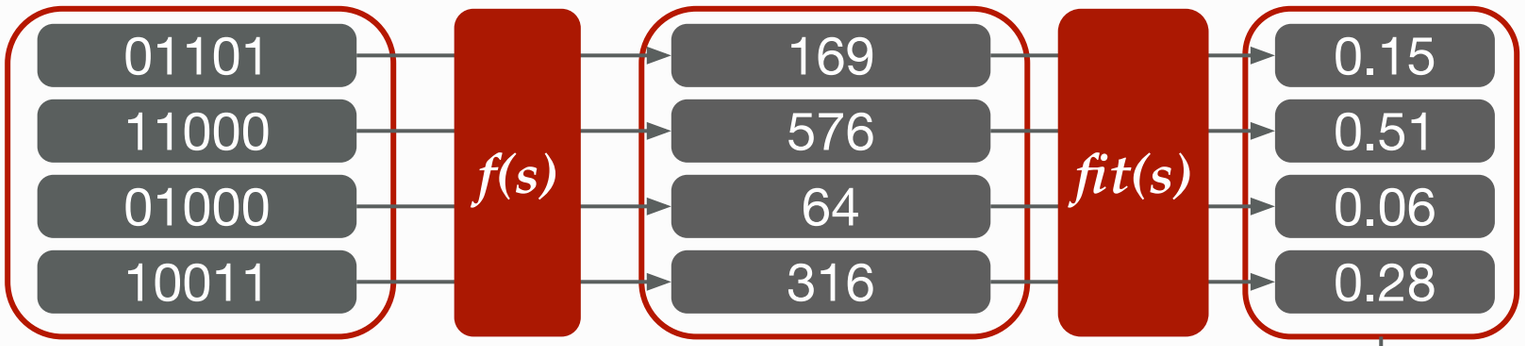


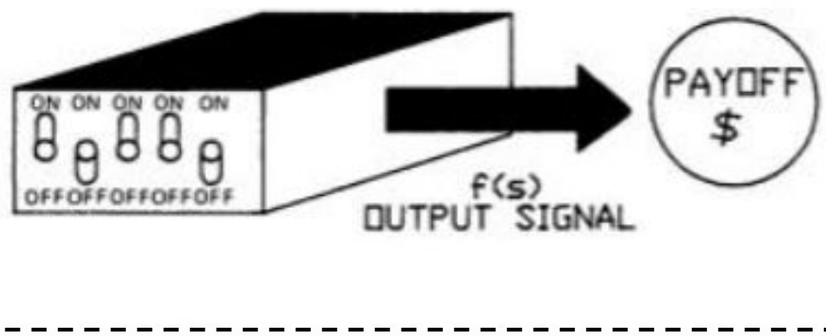
Algoritmi Genetici

Running Example

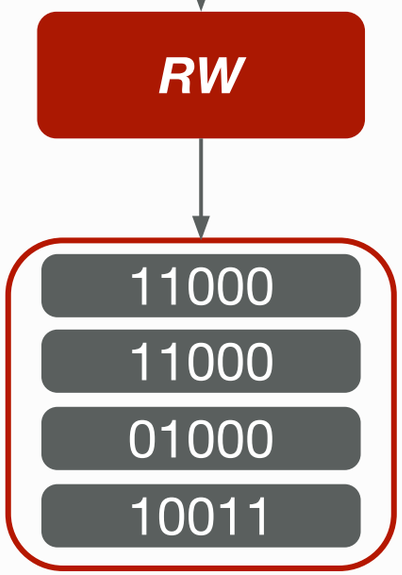
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**



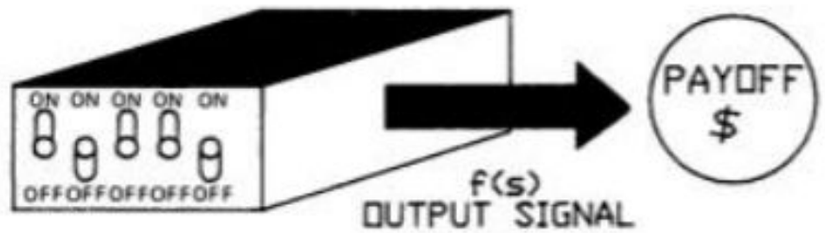
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

11000
11000
01000
10011



1. **Roulette Wheel Selection**

A questo punto, si creano abbinamenti casuali di tutti gli individui, così che ogni individui possieda un (e un solo) partner. Dopodiché, si itera su ogni coppia e, con probabilità p_c (arbitraria), si effettua la ricombinazione.

2. **Single-point Crossover**

Il single-point crossover consiste nella selezione casuale (distribuzione uniforme) di un **punto di taglio** all'interno della codifica dei genitori, così da incrociare le due parti e dare vita a due offspring.

3. **Bit-Flip Mutation**

Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

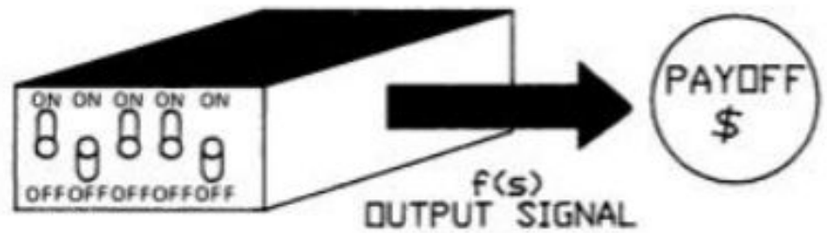
GA in azione

11000

11000

01000

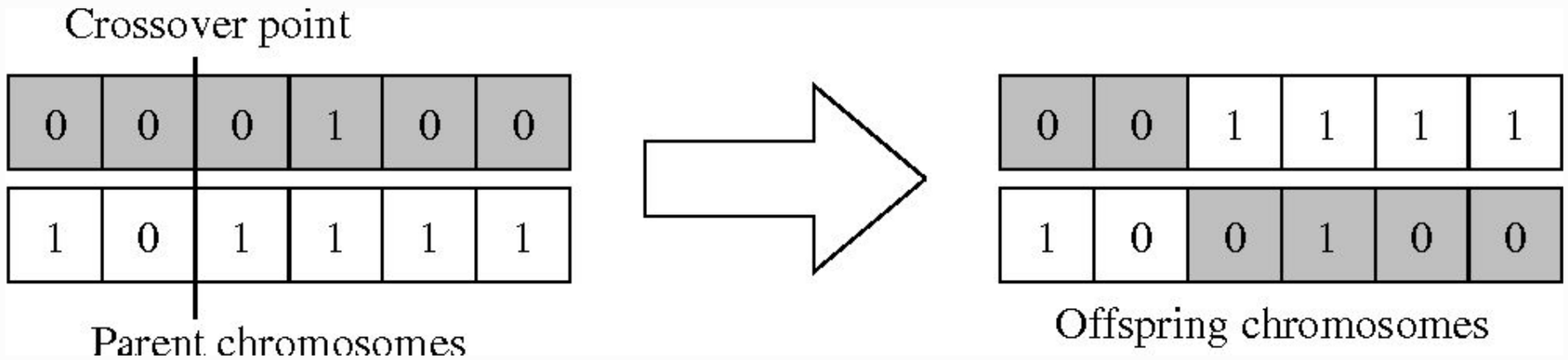
10011



1 **Roulette Wheel Selection**

2 **Single-point Crossover**

3 **Bit-Flip Mutation**

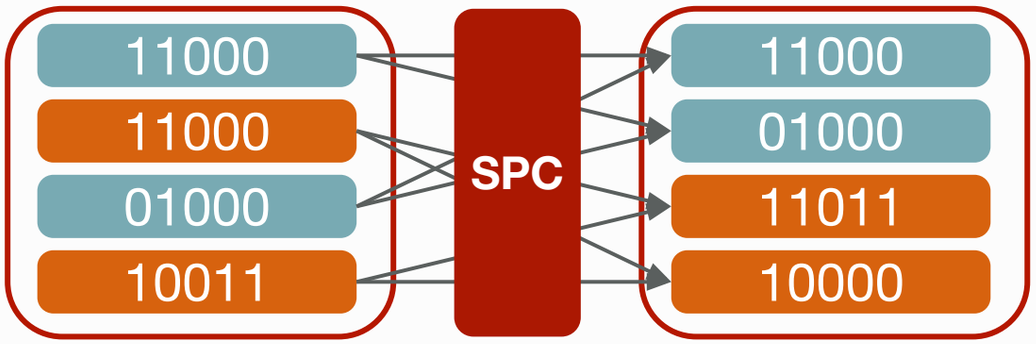


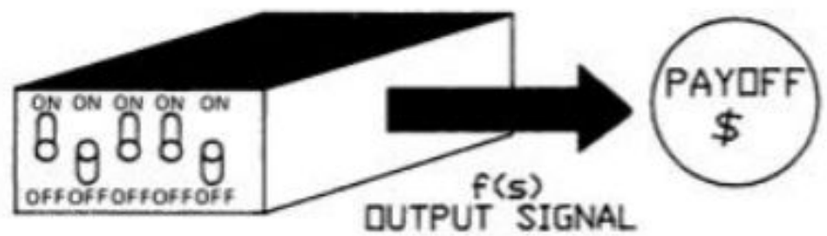
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

Con codifiche piccole non è raro avere figli molto simili, se non identici ai genitori...

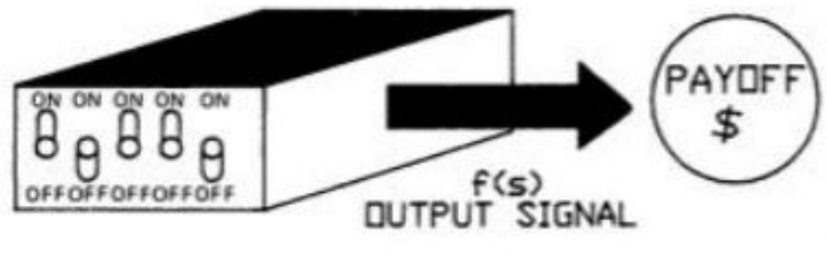
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

11000
11000
11011
10000



1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

Considerando tutti i geni presenti nella popolazione insieme (ovvero, 11000110001101110000), con probabilità p_m (arbitraria, di solito molto bassa), si applica una variazione del gene con un altro valore ammissibile. Avendo codifiche binarie, possiamo optare per un operazione di **bit flip**, ovvero l'inversione dello 0 con 1, e viceversa.

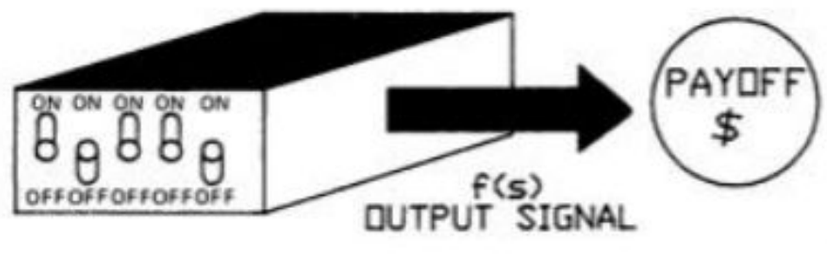
Algoritmi Genetici

Running Example

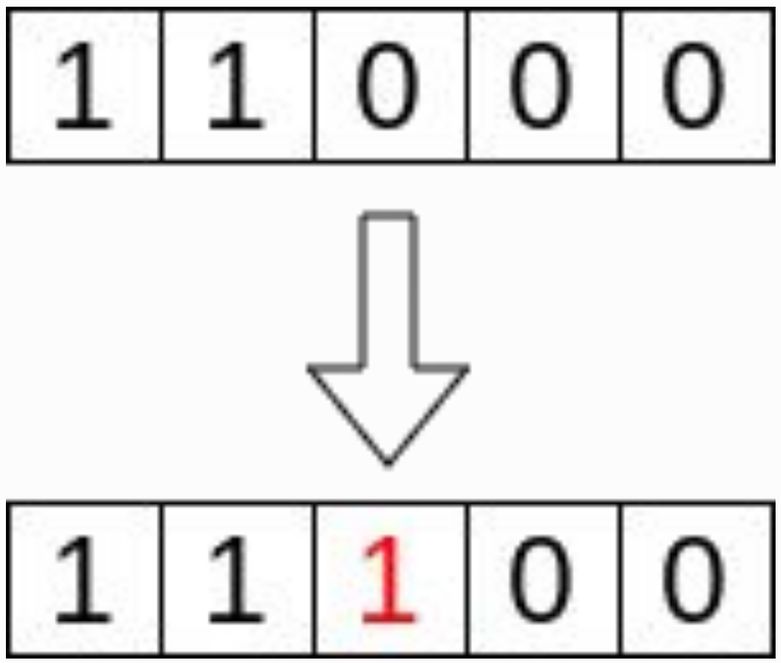
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione

11000
11000
11011
10000



1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

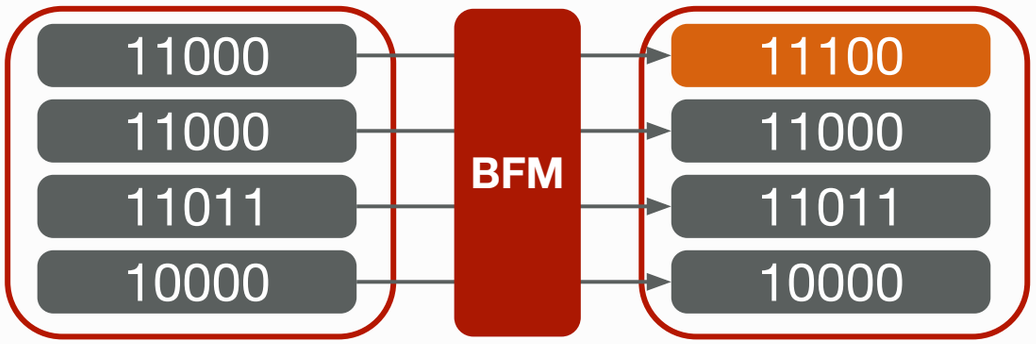


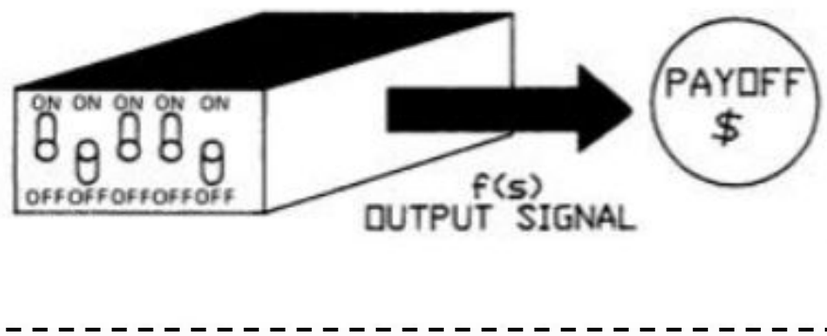
Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





1. **Roulette Wheel Selection**
2. **Single-point Crossover**
3. **Bit-Flip Mutation**

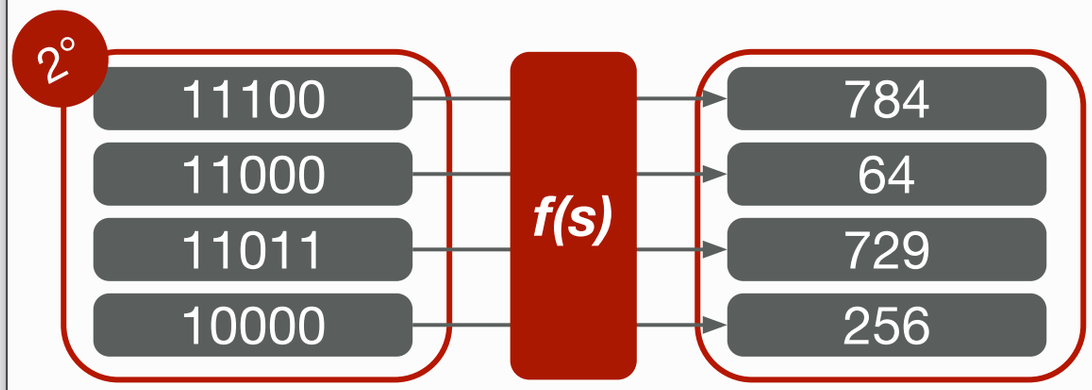
Abbiamo finalmente completato la creazione della nuova generazione di individui! Non resta altro che ripetere la valutazione (sottoponendo ciascun individuo alla funzione di valutazione).

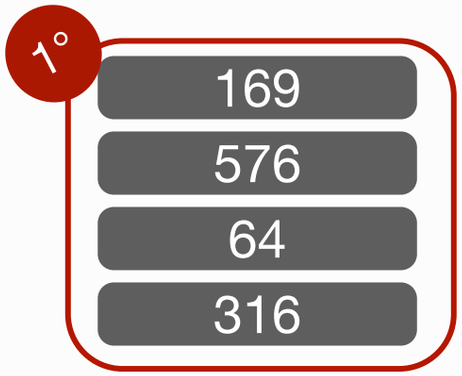
Algoritmi Genetici

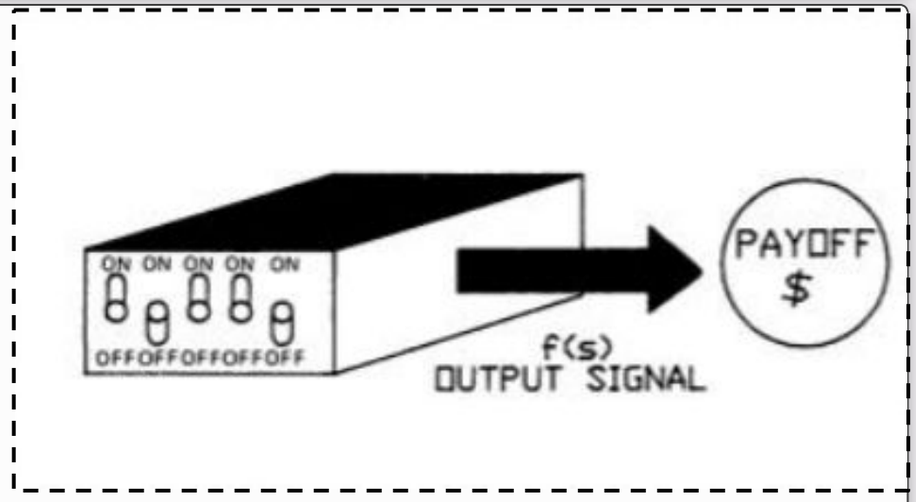
Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione







Rispetto alla generazione precedente abbiamo avuto dei sensibili miglioramenti: il miglior individuo ha avuto un miglioramento del 36% rispetto al miglior individuo della precedente iterazione.

Adesso ci si chiede se l'evoluzione debba terminare oppure proseguire. In altre parole: abbiamo raggiunto il **criterio di arresto**?

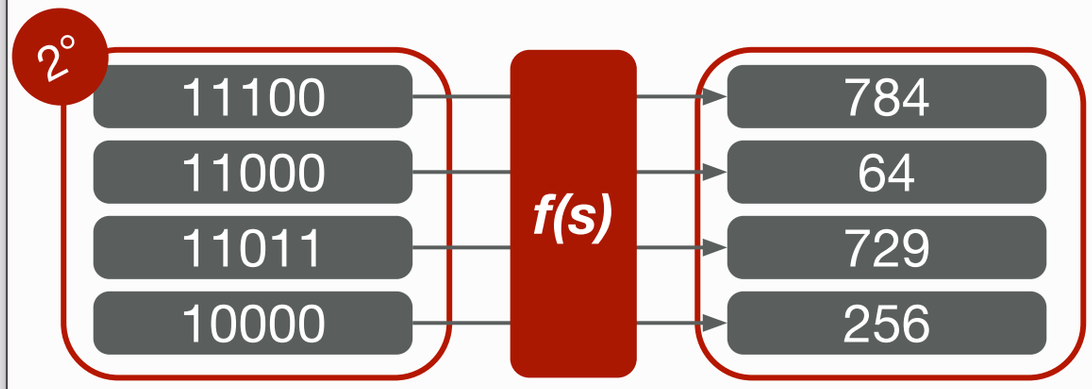
In questo esempio, adottiamo il seguente criterio: "se il miglior individuo non ha una valutazione superiore al migliore della generazione precedente, allora l'algoritmo termina".

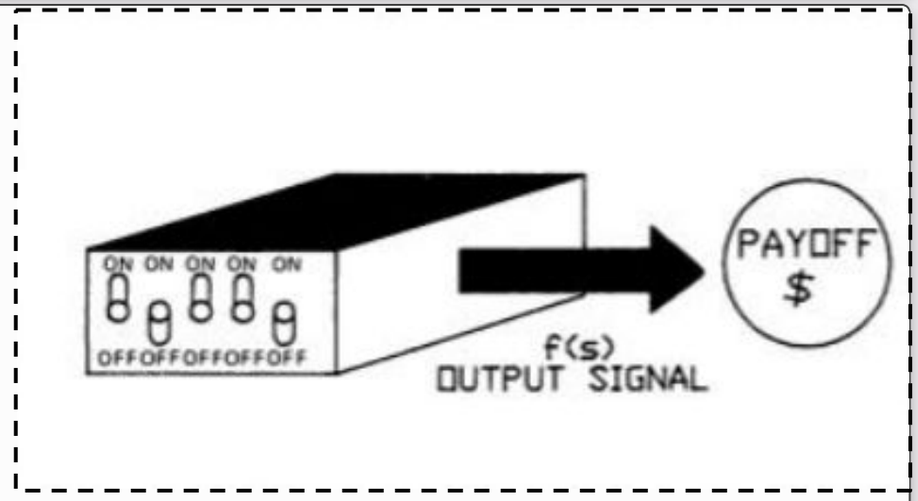
Algoritmi Genetici

Running Example

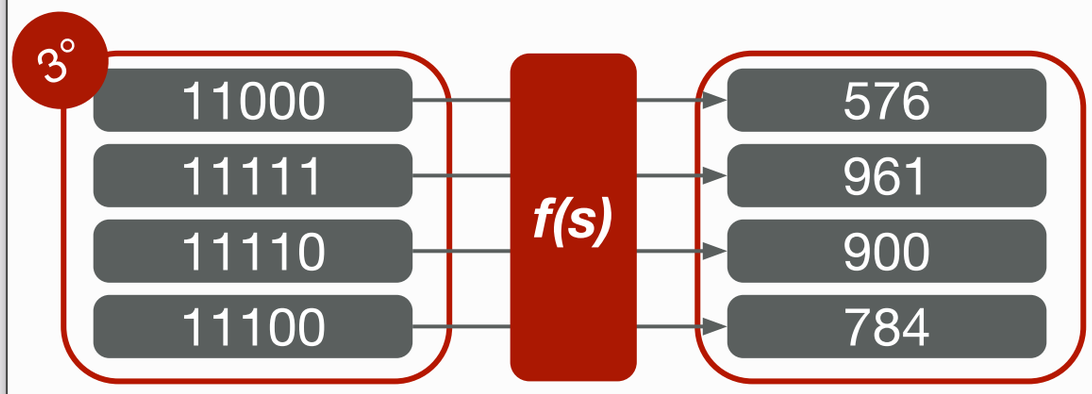
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





Quindi, possiamo proseguire con l'evoluzione...

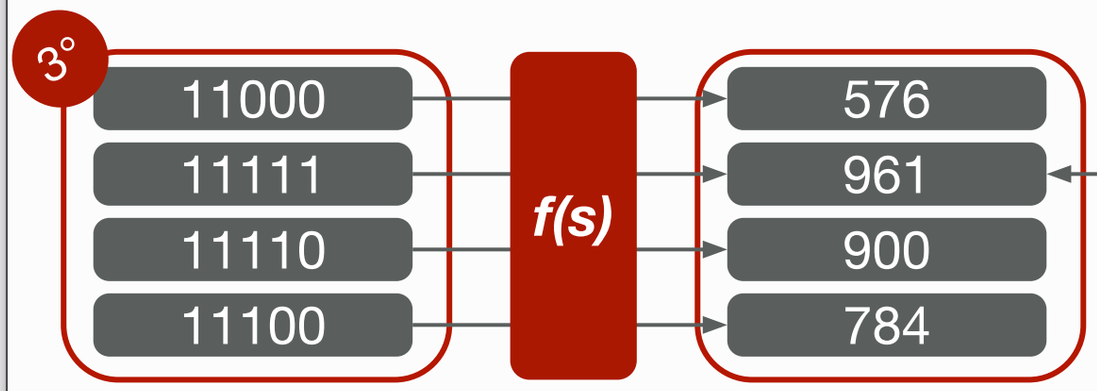


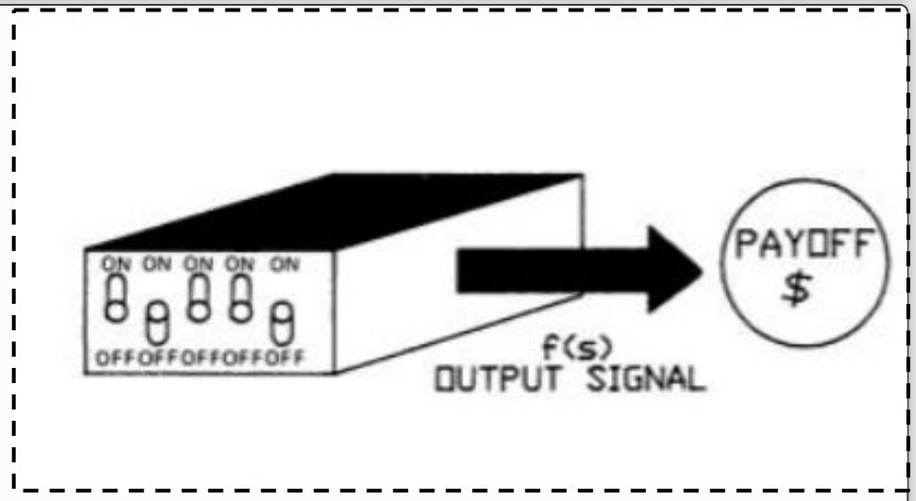
Algoritmi Genetici

Running Example

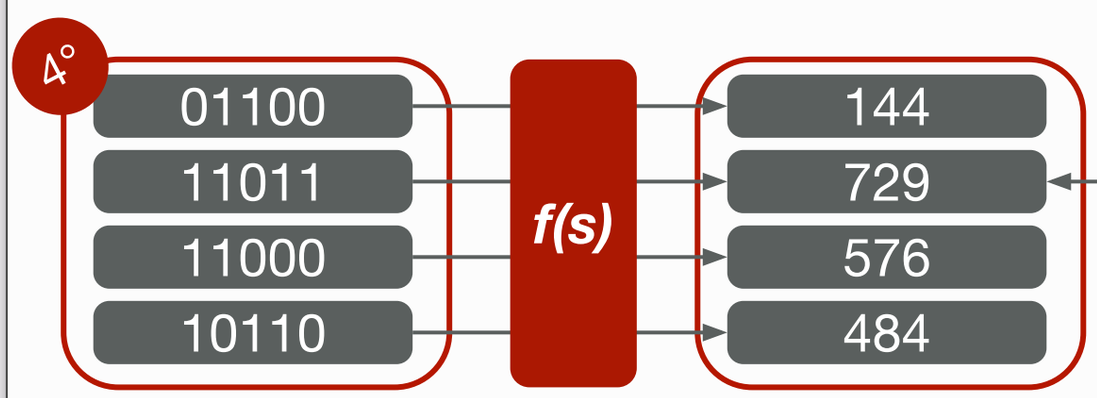
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





Quindi, possiamo proseguire con l'evoluzione...



Il criterio di arresto viene verificato poiché l'individuo 11011 non è migliore di 11111. L'evoluzione termina qui.

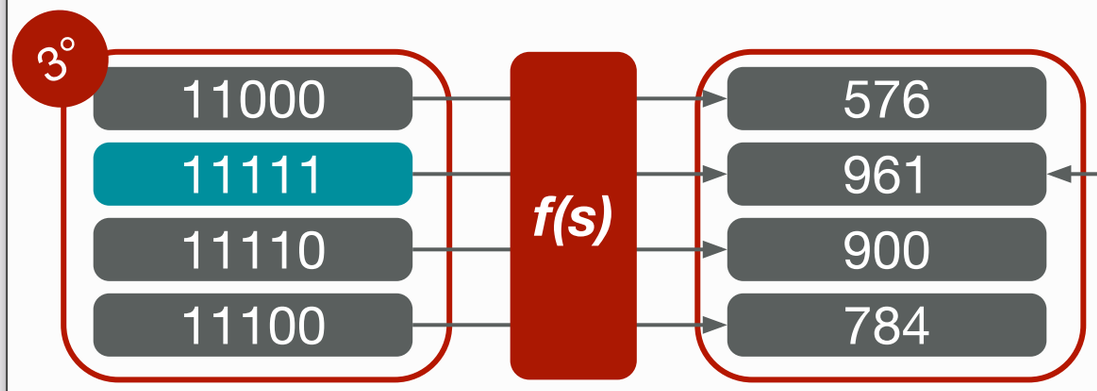
Cosa bisogna restituire? Con questo criterio di arresto non avrebbe senso restituire l'ultima generazione dato che la penultima risulta migliore. Quindi potremmo restituire l'**ultima migliore**.

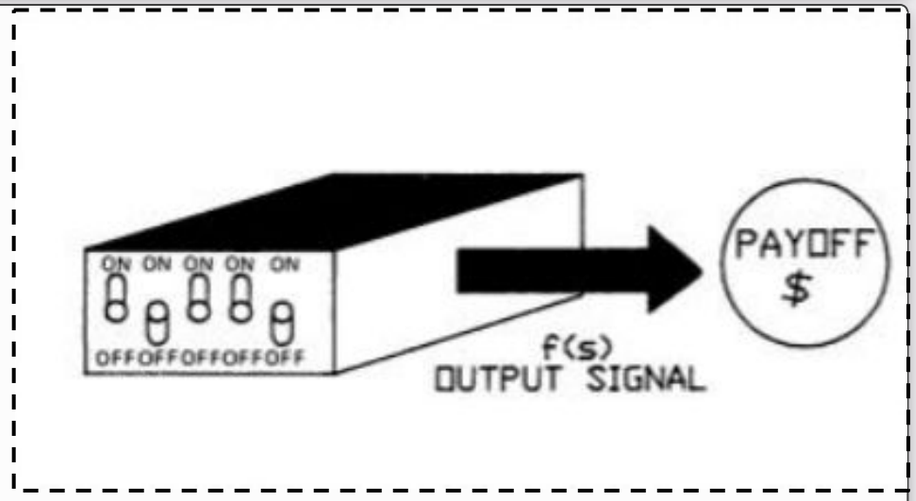
Algoritmi Genetici

Running Example

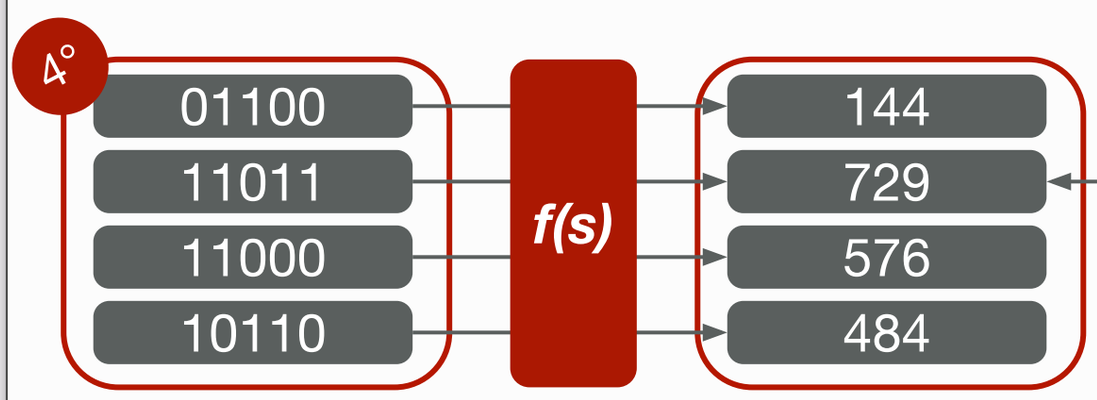
Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione





Quindi, possiamo proseguire con l'evoluzione...



Il criterio di arresto viene verificato poiché l'individuo 11011 non è migliore di 11111. L'evoluzione termina qui.

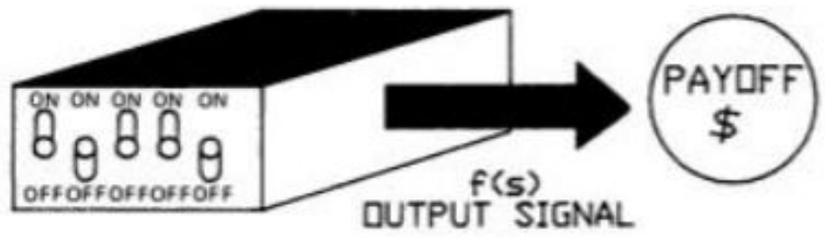
In particolare, possiamo restituire direttamente il **miglior individuo della migliore generazione**, quello con codifica 11111.

Algoritmi Genetici

Running Example

Consideriamo una scatola nera con 5 interruttori. Non sappiamo molto di questa scatola, soltanto che il suo comportamento è modellato dalla funzione $f(s)$, dove s è una configurazione dei 5 interruttori. La funzione restituisce un valore intero, il nostro obiettivo è **massimizzarla**, ovvero trovare la configurazione che dia il massimo di $f(s)$.

GA in azione - Considerazioni finali



Abbiamo ottenuto 11111, che sarebbe l'individuo che il GA considera sia il migliore. In realtà, dato non conosciamo niente della funzione obiettivo (ovvero, non disponiamo di un *test di ottimalità*) non possiamo dire con certezza che 11111 sia davvero il punto di ottimo globale.

In scenari totalmente black-box, quindi, non possiamo fare di meglio… però adesso possiamo scoprire le carte. In realtà $f(s) = x^2$. Quindi, l'individuo 11111 risulta effettivamente il punto di ottimo globale.

Algoritmi Genetici

Pseudocodice

```
function SIMPLE-GA(fitness_function, stopping_condition)
                          returns an individual
    P <- init_population()
    while stopping_condition is false
       evaluate(fitness_function, P)
       P_1 <- selection(P)
       P_2 <- crossover(P_1)
       P_3 <- mutation(P_2)
       P <- P_3
    b <- get_best_individual(fitness_function, P)
    return b
```

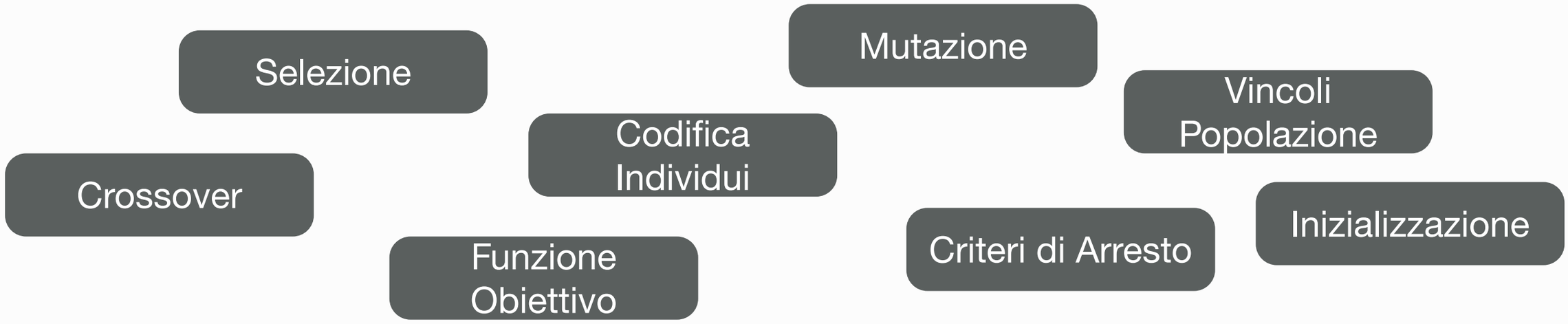
Questa è la forma più semplice di un algoritmo genetico di tipo **generazionale**: ad ogni iterazione si crea una nuova generazione. Un algoritmo di questo tipo è comunemente noto come **Simple Genetic Algorithm** (SGA).

Da un punto di vista operativo è sempre bene conservare la storia dell'intera evoluzione invece che rimpiazzare del tutto quella precedente. Quindi conviene mantenere una struttura dati che conservi lo storico di tutte (o almeno le ultime X) le generazioni.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:



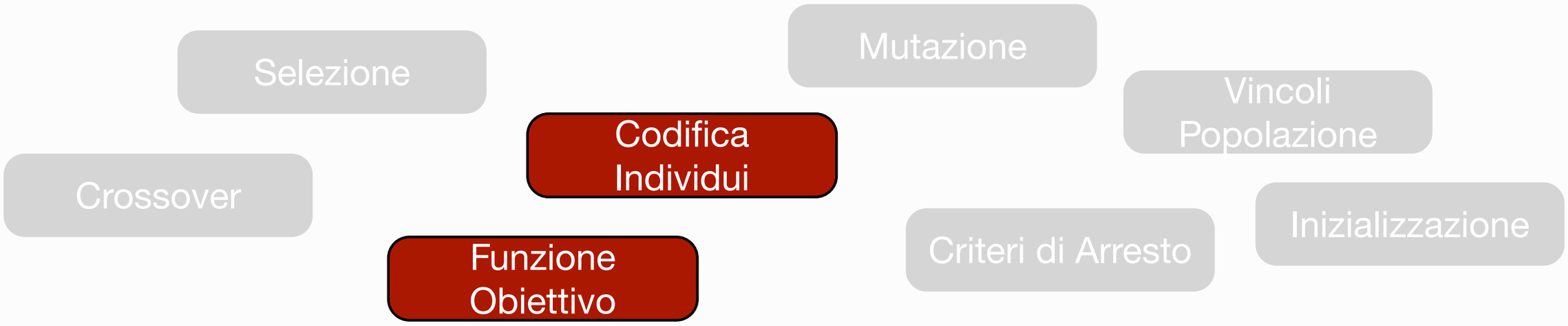
La vera forza dei GA risiede proprio in questo: invece di adattare uno specifico algoritmo di ottimizzazione per risolvere un problema di ottimizzazione, conviene adattare il problema alla meta-euristica. Il problema specifico, infatti, potrebbe avere tante particolarità difficili da affrontare, mentre le componenti di un GA, pur essendo tante, sono **semplici** da affrontare, e, soprattutto, **ricorrenti** tra tanti problemi.

Esistono molti algoritmi di selezione, crossover e mutazione ben noti e già pronti all'uso (ne vedremo alcuni), così come i criteri di arresto. La difficoltà maggiore risiede soltanto in due aspetti.

Algoritmi Genetici

Modellazione del Problema

I GA sono caratterizzati da molte componenti:



La vera forza dei GA risiede proprio in questo: invece di adattare uno specifico algoritmo di ottimizzazione per risolvere un problema di ottimizzazione, conviene adattare il problema alla meta-euristica. Il problema specifico, infatti, potrebbe avere tante particolarità difficili da affrontare, mentre le componenti di un GA, pur essendo tante, sono **semplici** da affrontare, e, soprattutto, **ricorrenti** tra tanti problemi.

Esistono molti algoritmi di selezione, crossover e mutazione ben noti e già pronti all'uso (ne vedremo alcuni), così come i criteri di arresto. La difficoltà maggiore risiede soltanto in due aspetti.

Una volta stabilita la funzione obiettivo e scelta un'opportuna codifica degli individui, abbiamo *praticamente già risolto il problema*. Resta soltanto "giocare" con le altre componenti, **sperimentando** diverse configurazioni per migliorare le prestazioni.

Algoritmi Genetici

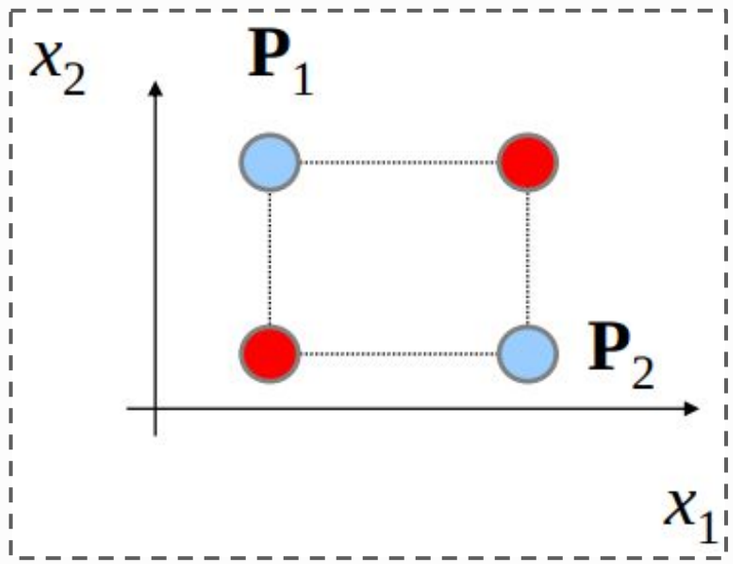
Codifica Individui

La scuola di Holland prevede che la codifica degli individui sia sempre una **stringa binaria** (come abbiamo visto nell'esempio). In generale, infatti, la codifica binaria è preferibile, poiché si presta bene agli operatori che abbiamo descritto. Ha, però, un grande problema: non scala bene. Per rappresentare soluzioni candidate "piccole" abbiamo bisogno di grandi codifiche. Questo problema si accentua considerando che noi facciamo evolvere centinaia/migliaia di individui per centinaia/migliaia di iterazioni.

Per quanto possibile, bisogna trovare codifiche **quanto più compatte possibile**, codificando soltanto le parti essenziali. Qualora non fosse possibile, si opta per altri tipi di codifiche, ad esempio:

- Stringhe di numeri reali (**real-coded GA**)
- Strutture dati (e.g., alberi, grafi)
- Altre strutture complesse (e.g., reti neurali, programmi)

La **codifica reale** è molto gettonata in quanto più compatta e più vicina al dominio di problema che prendono input numerici. In questo scenario i classici operatori di crossover e mutazione, per quanto comunque riusabili, creano meno diversificazione rispetto al caso binario. Per fare un esempio, non è difficile avere problemi con individui descritti da soli due numeri reali. Un single-point crossover risulterebbe poco diversificante, come si vede in figura.



Algoritmi Genetici

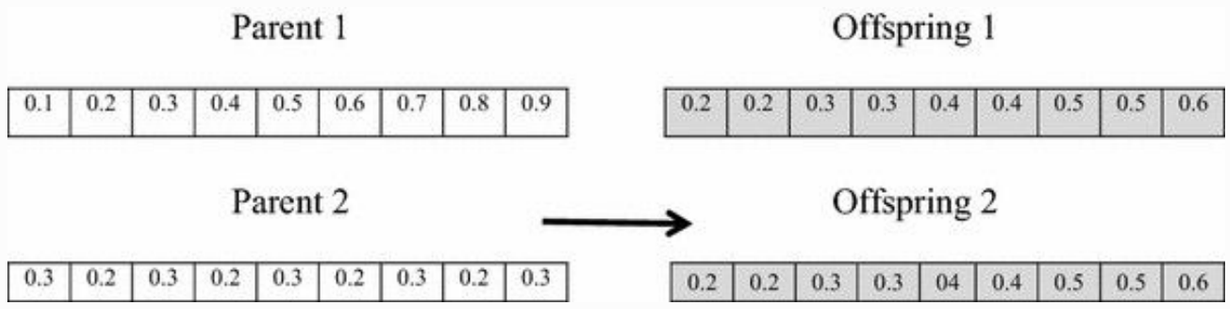
Codifica Individui

La scuola di Holland prevede che la codifica degli individui sia sempre una **stringa binaria** (come abbiamo visto nell'esempio). In generale, infatti, la codifica binaria è preferibile, poiché si presta bene agli operatori che abbiamo descritto. Ha, però, un grande problema: non scala bene. Per rappresentare soluzioni candidate "piccole" abbiamo bisogno di grandi codifiche. Questo problema si accentua considerando che noi facciamo evolvere centinaia/migliaia di individui per centinaia/migliaia di iterazioni.

Per quanto possibile, bisogna trovare codifiche **quanto più compatte possibile**, codificando soltanto le parti essenziali. Qualora non fosse possibile, si opta per altri tipi di codifiche, ad esempio:

- Stringhe di numeri reali (**real-coded GA**)
- Strutture dati (e.g., alberi, grafi)
- Altre strutture complesse (e.g., reti neurali, programmi)

Per questa ragione sono stati definiti operatori specifici per real-coded GA, ad esempio, l'**Arithmetic Crossover**. Dati due parent $X = <x_1, x_2, \ldots, x_n>$ e $Y = <y_1, y_2, \ldots y_n>$, il primo figlio risulta $O_1 = \alpha \cdot X + (1 - \alpha) \cdot Y$, con $0 \leq \alpha \leq 1$ (per il secondo figlio O_2 è analogo ma con i coefficienti scambiati). Con $\alpha = 0.5$ si ottengono due figli identici.



Chiaramente esistono altri operatori più specifici, che però tralasciamo.

Algoritmi Genetici

Codifica Individui - Un Esempio

Vediamo un esempio di codifica considerando il **problema delle 8 regine.**

Quale codifica si presterebbe meglio?

Una codifica che prevede una **stringa lunga 8 interi**, ciascuno che può assumere valore da 0 a 7. Ogni elemento rappresenta una colonna, mentre il valore corrisponde alla riga occupata dalla regina.

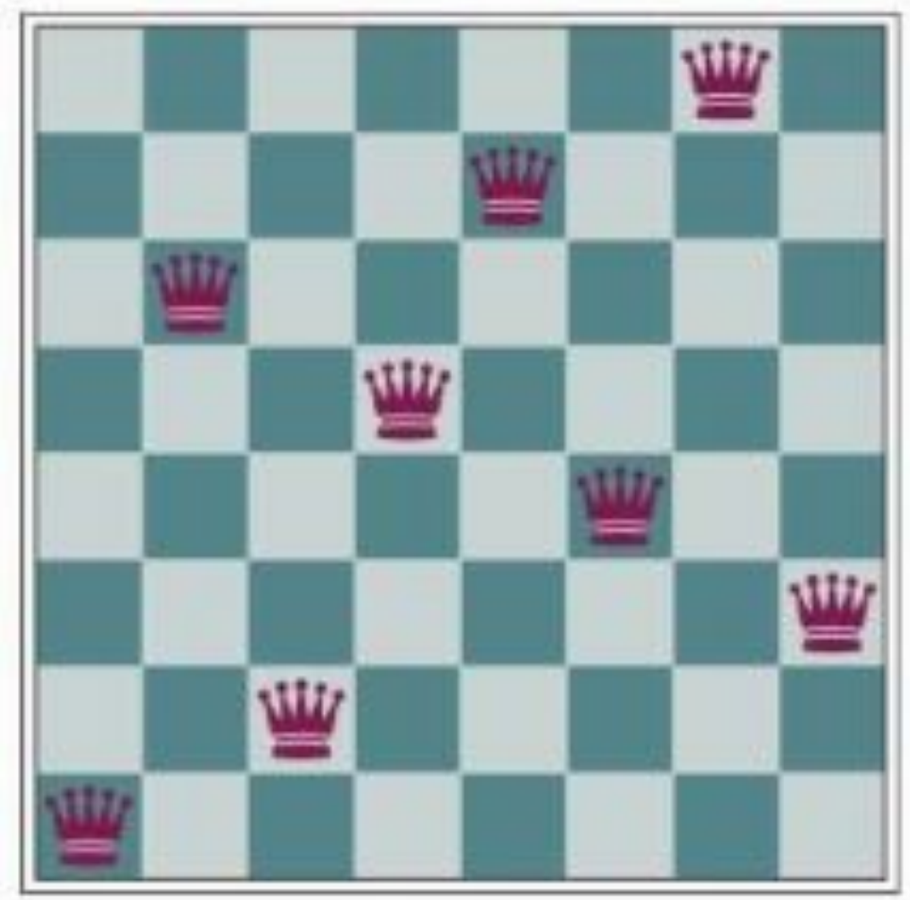
L'individuo in figura si codificherebbe nel seguente modo: $<7, 2, 6, 3, 1, 4, 0, 5>$

Chiaramente esistono altre codifiche, ma non tutte sono buone allo stesso modo.

La scelta della codifica ha un impatto nella scelta algoritmi di crossover e mutazione.

Quali operatori si presterebbero meglio?

Sicuramente un crossover aritmetico non ha senso perché darebbe luogo a codifiche fuori dal dominio. Considerata la natura del problema, ha senso riusare i crossover classici (e.g., il single-point crossover), che corrisponderebbe a creare nuove scacchiere incrociando le colonne. Mentre la mutazione andarebbe a modificare le righe occupate dalle regine.



Algoritmi Genetici

Funzione Obiettivo

Ricordiamo la differenza tra funzione obiettivo e funzione di valutazione:

- **Funzione obiettivo**: funzione matematica da ottimizzare che modella il problema.
- **Funzione di valutazione**: funzione che assegna un valore di bontà agli individui.

La funzione di valutazione è un concetto strettamente legato ai GAs, mentre la funzione obiettivo è legata al problema da risolvere. Le due **spesso coincidono** (si usa la funzione obiettivo anche come valutazione degli individui), tant'è che molti non fanno questa distinzione esplicita. Ciò non avviene quando **non possiamo** (o non vogliamo) usare direttamente la funzione obiettivo per ragioni di **performance o accessibilità**. Ove possibile, è bene valutare gli individui direttamente con la funzione obiettivo.

L'esempio giocattolo, in realtà, non rappresenta propriamente i **problemi del mondo reale** che incontriamo. Spesso, infatti, la funzione obiettivo NON è ancora già definita, ma **siamo noi progettisti a doverla definire in modo tale che modelli correttamente il problema da risolvere**.
Entro certi limiti, la scelta è "arbitraria", quindi siamo "liberi" di definirla come più ci fa comodo (chiaramente, dobbiamo modellare correttamente il problema).

Dobbiamo stabilire i valori che deve accettare (dominio), i valori che restituisce, l'andamento, ecc. Come possiamo sfruttare questa cosa a nostro vantaggio? Di che cosa abbiamo bisogno?

Algoritmi Genetici

Funzione Obiettivo

Se ricordiamo, l'esempio giocattolo non forniva un **test di ottimalità**, ovvero un meccanismo che ci permetta di capire se abbiamo trovato già l'ottimo globale, e quindi poter arrestare l'evoluzione prima del tempo.

Un trucco semplice potrebbe essere quello di definire la funzione in **forma di minimo**, e fare in modo che abbia **lower bound zero**. Ad esempio, la funzione potrebbe avere valori $[0, +\infty[$ oppure $[0, 1]$.

Se la funzione obiettivo che abbiamo modellato è eseguibile in tempi accettabili, allora possiamo anche riusarla come funzione di valutazione. Quindi, se durante l'evoluzione otteniamo un individuo valutato con punteggio 0, allora abbiamo trovato il punto di ottimo globale!

Se la funzione obiettivo che abbiamo modellato NON è eseguibile in tempi accettabili, allora possiamo definire la nostra funzione di valutazione che approssima/stima la funzione obiettivo, cercando di fare in modo che anch'essa abbia lower bound zero.

Algoritmi Genetici

Funzione Obiettivo - Un Esempio

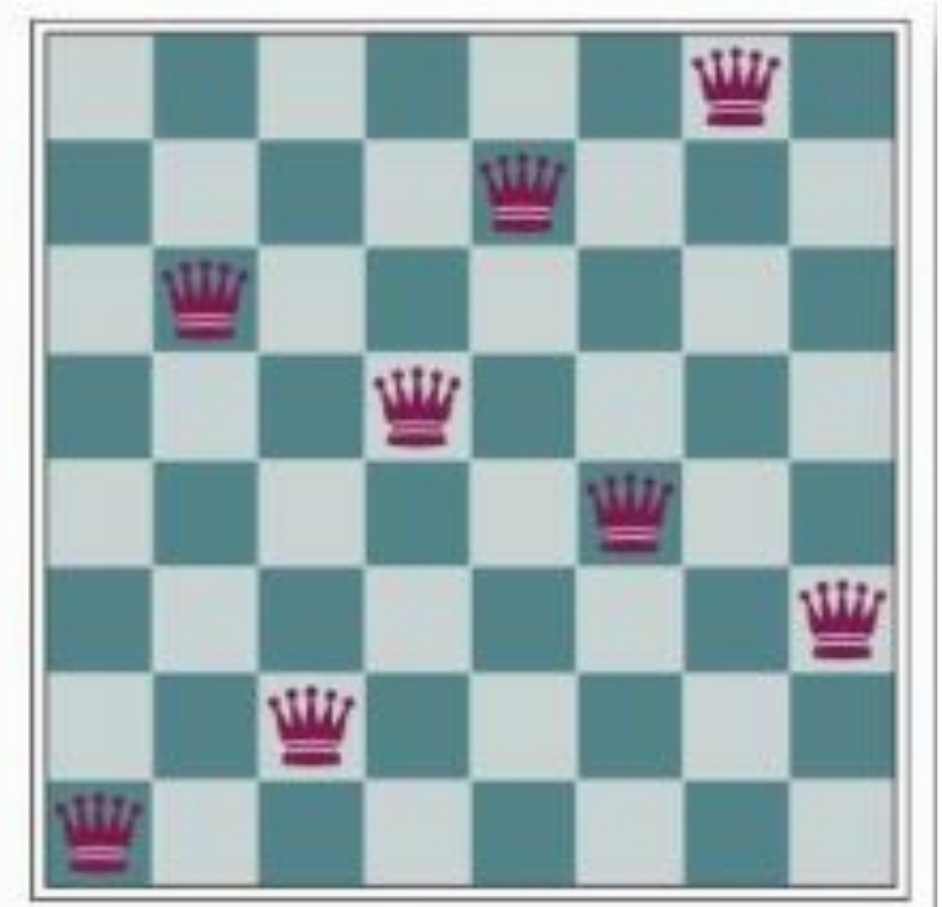
Ritorniamo al **problema delle 8 regine**.

Quale funzione obiettivo modella correttamente il problema?

Il nostro scopo è quello di *minimizzare il numero di* conflitti. Quindi ha senso che la funzione obiettivo sia in grado di catturare il numero di conflitti. Banalmente, possiamo definirla come un **conteggio del numero di conflitti**.

Con la codifica definita prima, il conteggio è abbastanza semplice da calcolare, quindi possiamo riusare la stessa funzione obiettivo per valutare gli individui.

Essendo il conteggio una funzione con lower bound zero, allora disponiamo di un semplice test di ottimalità: se un individuo ha zero conflitti, è ottimale (banalmente).



Algoritmi Genetici

GAs: Croce e Delizia

Flessibili

I GAs possono essere visti come dei *framework* in grado di risolvere un'**ampia gamma di problemi**. Due sono i punti cruciali da affrontare: codifica delle soluzioni candidate, e modellazione della funzione obiettivo. Se questi due task sono risolti facilmente, allora i GA sono una buona scelta.

Esplorazione Rapida

La loro capacità di compiere ricerca globale permette una rapida esplorazione dello spazio di ricerca, evitando in poco tempo aree poco promettenti (ovvero, ben lontane dall'ottimo). Sono in grado di restituire **risultati accettabili** in poco tempo.

Continuo e Discreto

I GAs sono in grado di ottimizzare sia funzioni nel continuo che nel discreto, e si prestano bene quando abbiamo funzioni che accettano input con molte variabili (alta dimensionalità) o comunque strutturati in modo complesso, dove gli altri algoritmi di ottimizzazioni falliscono.

Parametri

Ci sono tantissimi componenti e parametri da decidere, e non esiste la combinazione ottimale nota a priori e generale (No Free Lunch Theorem). Quindi, è necessario fare **valutazioni empiriche** delle combinazioni che si reputano più promettenti. Conoscere al meglio il problema modellato permette di scartare alcune combinazioni poco promettenti.

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Laurea triennale in Informatica
Anno accademico 2021/2022

Fondamenti di Intelligenza Artificiale

Lezione 7 - Algoritmi di ricerca locale (II)