



Algoritmi genetici ed evolutivi

Giulia Cuttone

*Tutor: Professore
Giosue' Lo Bosco*

COMPLESSITÀ DEI PROBLEMI

- I problemi **NP-difficili** (**NP-hard problem**, da nondeterministic polynomial-time hard problem) sono una classe di problemi che può essere definita informalmente come la classe dei problemi non meno difficili dei problemi **NP-completi**, che a loro volta sono per definizione i più difficili delle classi **P/NP**.
- Più formalmente, un problema **H** è NP-difficile se e solo se ogni problema NP **L** è polinomialmente riducibile ad **H**, ovvero tale che $L \leq_T H$, $\forall L \in NP$.
- In altre parole, **L** deve poter essere risolto in tempo polinomiale da una macchina di Turing non deterministica dotata di un oracolo per **H**.
- La categoria dei problemi NP-difficili, a differenza delle classi P, NP e degli NP-completi, non è limitata per definizione ai soli problemi di decisione; vi appartengono infatti anche problemi di ottimizzazione e di altri generi.

EURISTICHE PER L'OTTIMIZZAZIONE

- L'euristica è un approccio di risoluzione dei problemi molto diffuso nella **simulazione** per vari motivi tra cui:
 - la risoluzione ottimale del problema può essere impossibile;
 - la risoluzione ottimale del problema può essere troppo costosa in termini di tempo o di capacità di elaborazione.

POSSIBILI EURISTICHE PER L'OTTIMIZZAZIONE

1. Basati su singola soluzione:

- Algoritmo di salita
- Algoritmo simulated annealing

2. Basati su popolazione:

- Algoritmi evolutivi

ALGORITMO DI SALITA (ITERATED HILLCLIMBER)

- Esistono alcune versioni di **algoritmi di salita**. Differiscono nel modo in cui viene selezionata una nuova soluzione per il confronto con la soluzione corrente. Una versione di un semplice algoritmo (iterato) di salita (iterazioni MAX) è

```
begin
  t = 0
  repeat
    local = FALSE
    select a current solution  $v_c$  at random
    evaluate  $v_c$ 
    repeat
      select  $n$  new solutions in the neighborhood of  $v_c$ 
      select the solution  $v_n$  from the set of new solutions with the largest value of objective function  $f$ 
      if  $f(v_c) < f(v_n)$ 
        then  $v_c = v_n$ 
      else local = TRUE
    until local
    t = t + 1
  until t = MAX
end
```


ALGORITMO SIMULATED ANNEALING

- Il concetto di annealing ("ricottura") deriva dalla scienza dei metalli, dov'è usato per descrivere il processo di eliminazione di difetti reticolari dai cristalli tramite riscaldamento seguito da lento raffreddamento.
- In questo caso un difetto reticolare corrisponde ad una combinazione errata di due oggetti.

- La struttura della **procedura di simulated annealing** è

```
begin  
   $t = 0$   
  initialize temperature  $T$   
  select a current solution  $v_c$  at random  
  evaluate  $v_c$   
  repeat  
    repeat  
      select a new solution  $v_n$  in the neighborhood of  $v_c$   
      if  $f(v_c) < f(v_n)$   
        then  $v_c = v_n$   
      else if  $\text{random}[0,1) < \exp\{(f(v_n) - f(v_c))/T\}$   
        then  $v_c = v_n$   
      until (termination-condition)  
       $T = g(T, t)$   
       $t = t + 1$   
    until (stop-criterion)  
end
```

- La (condizione di terminazione) verifica se viene raggiunto l' "*equilibrio termico*", cioè se la distribuzione di probabilità delle nuove stringhe selezionate si avvicina alla **distribuzione di Boltzmann**.
- Tuttavia, in alcune implementazioni, questo ciclo di ripetizione viene eseguito solo k volte (k è un parametro aggiuntivo del metodo).
- La **temperatura T** si abbassa gradualmente ($g(T, t) < T$ per ogni t).
- L'algoritmo termina per un **piccolo valore di T** : il (criterio di arresto) controlla se il sistema è '*congelato*', cioè praticamente non vengono più accettate modifiche.

ALGORITMI EVOLUTIVI

- Usiamo un termine comune, **Algoritmi Evolutivi (EA)**, per tutti gli algoritmi ispirati dal meccanismo di evoluzione delle specie

Struttura di
algoritmo
evolutivo




```
begin
  t = 0
  initialize P(t)
  evaluate P(t)
  while (not termination-condition) do
    begin
      t = t+1
      select P(t) from P(t-1)
      alter P(t)
      evaluate P(t)
    end
  end
```


ALGORITMI GENETICI

- Gli algoritmi genetici sono particolari algoritmi evolutivi, caratterizzati da specifici operatori di combinazione tra soluzioni della popolazione
- Durante l'iterazione t , un algoritmo genetico mantiene una popolazione di potenziali soluzioni (cromosomi, vettori), $P(t) = \{x_1^t, \dots, x_n^t\}$.
- Ogni soluzione x_i^t viene valutata per dare una certa misura della sua "idoneità". Quindi, una nuova popolazione (**iterazione $t+1$**) viene formata selezionando gli individui più adatti (**selection step**).
- Alcuni membri di questa nuova popolazione subiscono alterazioni (**alter step**) per mezzo di operatori genetici, quali ad esempio **crossover** e **mutazione**, per formare nuove soluzioni.

- **Crossover** combina le caratteristiche di due cromosomi parentali in modo da generare uno o più figli.
 - Per esempio, se le soluzioni sono rappresentate da vettori a cinque elementi, allora considerati i genitori $(a_1, a_2, a_3, a_4, a_5)$ e $(b_1, b_2, b_3, b_4, b_5)$, il *crossover a singolo taglio* produrrebbe la prole $(a_1, a_2, b_3, b_4, b_5)$ e $(b_1, b_2, a_3, a_4, a_5)$, considerato il taglio in posizione 3.
- La **mutazione** altera arbitrariamente uno o più geni di un cromosoma selezionato, mediante un cambiamento casuale con una probabilità pari al tasso di mutazione.

- 
- The slide features a dark gray background with decorative light blue circuit-like lines. These lines, consisting of straight segments and small circles at junctions, are positioned along the left and right edges of the slide, framing the central text area.
- Un algoritmo genetico (come qualsiasi programma di evoluzione) per un particolare problema deve avere le seguenti cinque componenti:
 - una rappresentazione genetica per potenziali soluzioni al problema;
 - un modo per creare una popolazione iniziale di potenziali soluzioni;
 - una funzione di valutazione che riveste il ruolo dell'ambiente, valutando le soluzioni in termini di "idoneità";
 - operazioni genetiche che alterano la composizione dei figli;
 - valori per vari parametri utilizzati dall'algoritmo genetico (dimensione della popolazione, probabilità di applicazione di operatori genetici, ecc.)

ESEMPI DI PROBLEMI COMPLESSI

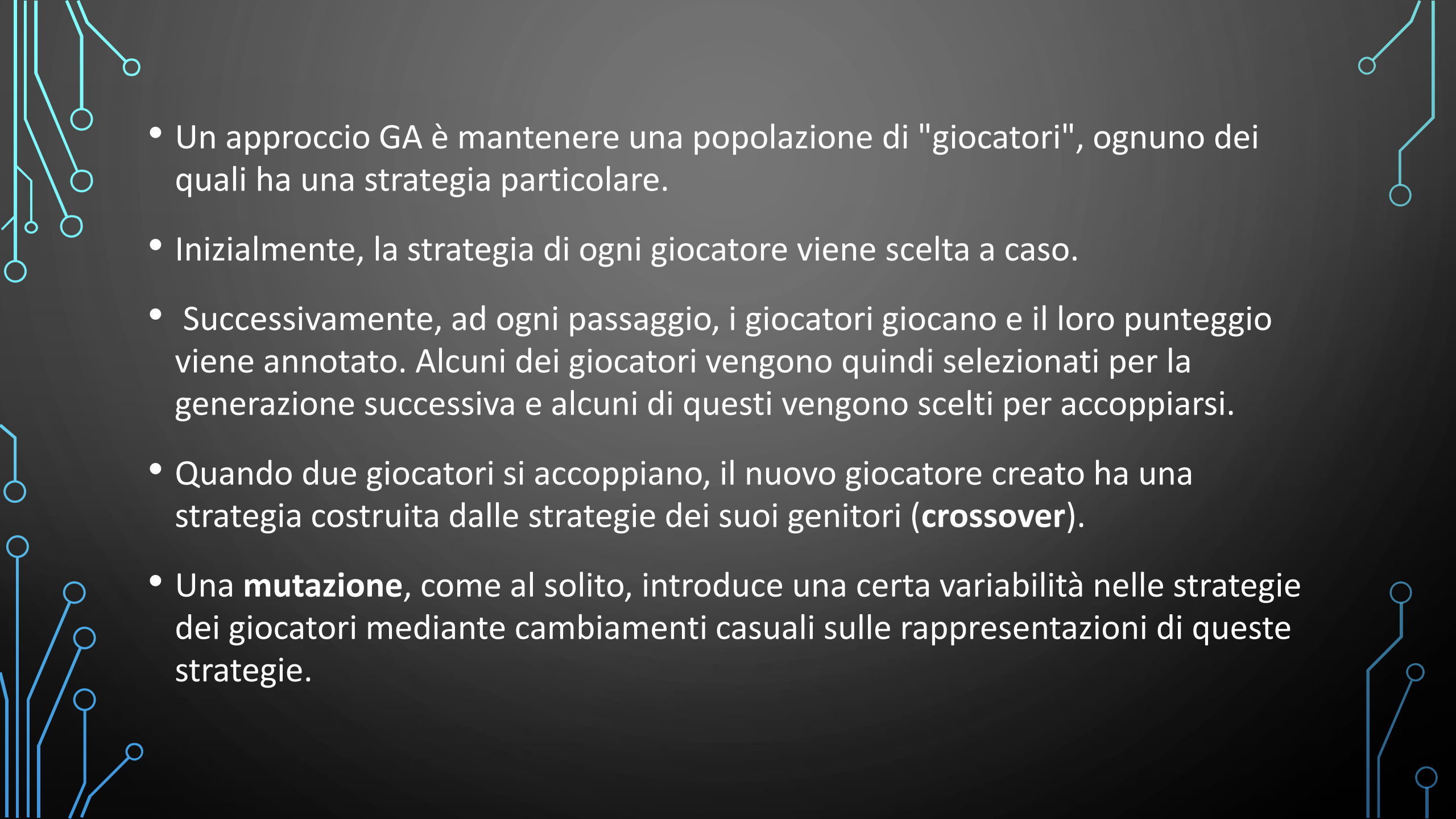
1. Il dilemma del prigioniero
2. Problema del commesso viaggiatore (TSP)

IL DILEMMA DEL PRIGIONIERO

- Due prigionieri sono tenuti in celle separate, incapaci di comunicare tra loro. A ogni prigioniero viene chiesto, in modo indipendente, di disertare e tradire l'altro prigioniero.
- Abbiamo tre scenari:
 - Se solo un prigioniero diserta, viene premiato e l'altro viene punito.
 - Se entrambi disertano, entrambi rimangono imprigionati e vengono torturati.
 - Se nessuno dei due diserta, entrambi ricevono ricompense moderate.

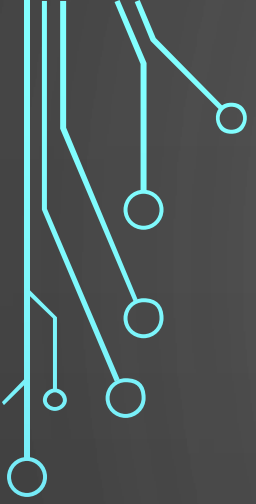

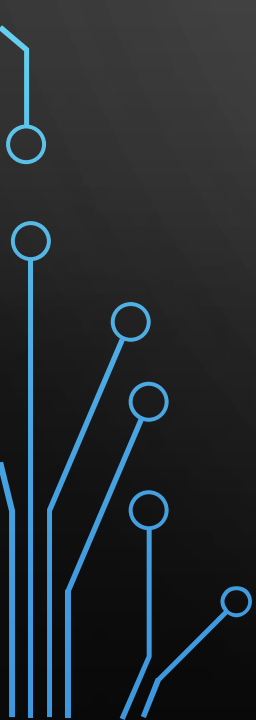
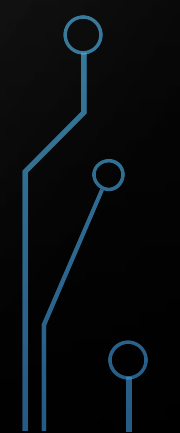
- Pertanto, la scelta egoistica della defezione produce sempre una ricompensa maggiore rispetto alla cooperazione - non importa cosa fa l'altro prigioniero - ma se entrambi disertano, entrambi fanno la scelta peggiore che se entrambi avessero collaborato.
- Il dilemma del prigioniero è decidere se disertare o cooperare con l'altro prigioniero; può essere visto come un gioco tra due giocatori, dove ad ogni turno, ogni giocatore diserta o collabora con l'altro prigioniero.
 - Esempio: Le vincite sono elencate nella seguente tabella

Giocatore 1	Giocatore 2	G1	G2	Commento
Diserta	Diserta	1	1	Punizione per defezione reciproca
Diserta	Collabora	5	0	Tentativo di defezione con ricompensa, punizione
Collabora	Diserta	0	5	Punizione, tentativo di defezione con ricompensa
Collabora	Collabora	3	3	Ricompensa per la cooperazione reciproca

- 
- The slide features a dark gray background with decorative light blue circuit-like lines in the corners. These lines consist of vertical and horizontal segments connected by small circles, resembling a stylized electronic circuit or a network diagram. The lines are positioned in the top-left, top-right, bottom-left, and bottom-right corners, framing the central text area.
- Un approccio GA è mantenere una popolazione di "giocatori", ognuno dei quali ha una strategia particolare.
 - Inizialmente, la strategia di ogni giocatore viene scelta a caso.
 - Successivamente, ad ogni passaggio, i giocatori giocano e il loro punteggio viene annotato. Alcuni dei giocatori vengono quindi selezionati per la generazione successiva e alcuni di questi vengono scelti per accoppiarsi.
 - Quando due giocatori si accoppiano, il nuovo giocatore creato ha una strategia costruita dalle strategie dei suoi genitori (**crossover**).
 - Una **mutazione**, come al solito, introduce una certa variabilità nelle strategie dei giocatori mediante cambiamenti casuali sulle rappresentazioni di queste strategie.

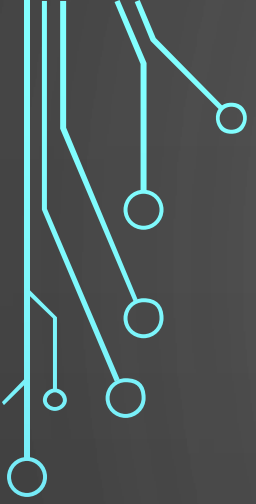

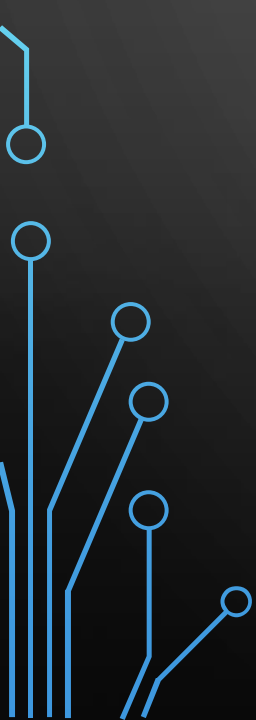
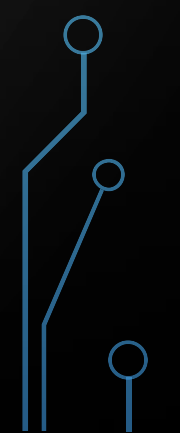
RAPPRESENTARE UNA STRATEGIA

- Prima di tutto, abbiamo bisogno di un modo per rappresentare una strategia (cioè una possibile soluzione).
- Per semplicità, prenderemo in considerazione strategie *deterministiche* e utilizzeremo i risultati delle tre mosse precedenti per fare una scelta nella mossa corrente.
- Poiché ci sono *quattro possibili risultati per ogni mossa*, ci sono **$4 \times 4 \times 4 = 64$** storie diverse di tre mosse precedenti.

- 
- 
- 
- 
- Per far partire la strategia all'inizio del gioco, dobbiamo anche specificare le sue premesse iniziali sulle tre mosse ipotetiche che hanno preceduto l'inizio del gioco.
 - Ciò richiede altri *sei geni*, per un totale di *settanta bit sul cromosoma*.
 - Questa *stringa di settanta bit* specifica cosa farebbe il giocatore in ogni possibile circostanza e quindi definisce completamente una particolare strategia.
 - La stringa di **70 geni** si riserva anche come cromosoma del giocatore per l'uso nel processo di evoluzione.

SCHEMA DELL'ALGORITMO GENETICO

- L'**algoritmo genetico di Axelrod** per apprendere una strategia per il dilemma del prigioniero segue quattro fasi:
 1. Si sceglie una popolazione iniziale. Ad ogni giocatore viene assegnata una **stringa casuale di settanta bit**, che rappresenta una strategia
 2. Si prova ogni giocatore per determinarne l'efficacia. Ogni giocatore usa la strategia definita dal suo cromosoma per giocare con gli altri giocatori. Il punteggio del giocatore è la *sua media* su tutte le partite che gioca
 3. Si selezionano i giocatori da allevare. Ad un giocatore con un *punteggio medio* viene assegnato un accoppiamento; un giocatore che segna una *deviazione standard sopra la media* riceve due accoppiamenti; e un giocatore che segna una *deviazione standard al di sotto della media* non riceve accoppiamenti
 4. I giocatori vincenti vengono accoppiati casualmente per produrre due figli per accoppiamento. La strategia di ogni prole è determinata dalle strategie dei suoi genitori. Questo viene fatto utilizzando due operatori genetici: **crossover** e **mutazione**

- 
- 
- 
- 
- Dopo queste quattro fasi otteniamo una nuova popolazione.
 - La nuova popolazione mostrerà comportamenti più simili a quelli degli individui di successo della generazione precedente e meno simili a quelli di quelli che non hanno avuto successo.
 - Con ogni nuova generazione, gli individui con punteggi relativamente alti avranno maggiori probabilità di trasmettere parti delle loro strategie, mentre gli individui relativamente senza successo avranno meno probabilità di avere parti delle loro strategie trasmesse.

RISULTATI SPERIMENTALI

- Eseguendo questo programma, Axelrod ha ottenuto risultati davvero notevoli.
- Da un inizio rigorosamente casuale, l'algoritmo genetico ha evoluto popolazioni il cui membro mediano ha avuto lo stesso successo dell'algoritmo euristico più noto. Alcuni modelli comportamentali si sono evoluti nella stragrande maggioranza degli individui; questi sono:
 - 1. Non scuotere la barca:** continua a cooperare dopo tre cooperazioni reciproche
(C dopo (CC) (CC) (CC))
 - 2. Sii provocatorio:** diserta quando l'altro giocatore diserta di punto in bianco
(D dopo aver ricevuto (CC) (CC) (CD))
 - 3. Accetta le scuse:** continua a collaborare dopo che la cooperazione è stata ripristinata
(C dopo (CD) (DC) (CC))
 - 4. Dimentica:** collabora quando la cooperazione reciproca è stata ripristinata dopo uno sfruttamento
(C dopo (DC) (CC) (CC))
 - 5. Accetta un solco:** diserta dopo tre defezioni reciproche
(D dopo (DD) (DD) (DD))

PROBLEMA DEL COMMESSE VIAGGIATORE (TSP)

- Il commesso viaggiatore deve visitare tutte le città del suo territorio esattamente una volta e poi tornare al punto di partenza; dato il costo del viaggio tra coppie di città, come dovrebbe pianificare il suo itinerario per il costo minimo totale dell'intero tour?
- Il TSP è un problema nell'ottimizzazione combinatoria e si presenta in numerose applicazioni.
- Ci sono diversi algoritmi branch-and-bound, algoritmi approssimativi e algoritmi di ricerca euristica che affrontano questo problema.

- Nell'esempio precedente (dilemma del prigioniero) abbiamo rappresentato un cromosoma (in modo più o meno naturale) come vettore binario. Questo non è il caso del problema del commesso viaggiatore.
- Sembra che la rappresentazione del **vettore intero** sia migliore
 - Un vettore $\mathbf{v} = \langle i_1 i_2 \dots i_n \rangle$ rappresenta un tour: da i_1 a i_2 , ..., da i_{n-1} a i_n e di nuovo a i_1 (\mathbf{v} è una **permutazione di $\langle 1 2 \dots n \rangle$**).
- Per il processo di inizializzazione possiamo utilizzare alcune *euristiche* (ad esempio, possiamo accettare alcuni output da un algoritmo **greedy** per il TSP, partendo da città diverse), oppure possiamo inizializzare la popolazione da un campione casuale di permutazioni di **$\langle 1 2 \dots n \rangle$** .

- Quindi ciò di cui abbiamo bisogno è un operatore simile a un crossover che sfrutti importanti somiglianze tra i cromosomi. A tale scopo utilizziamo una variante dell'**operatore OX** (crossover senza ripetizioni), che, dati due genitori, costruisce la prole scegliendo una sottosequenza di un tour da un genitore e preservando l'ordine relativo delle città dall'altro genitore.

- Per esempio, se i genitori sono

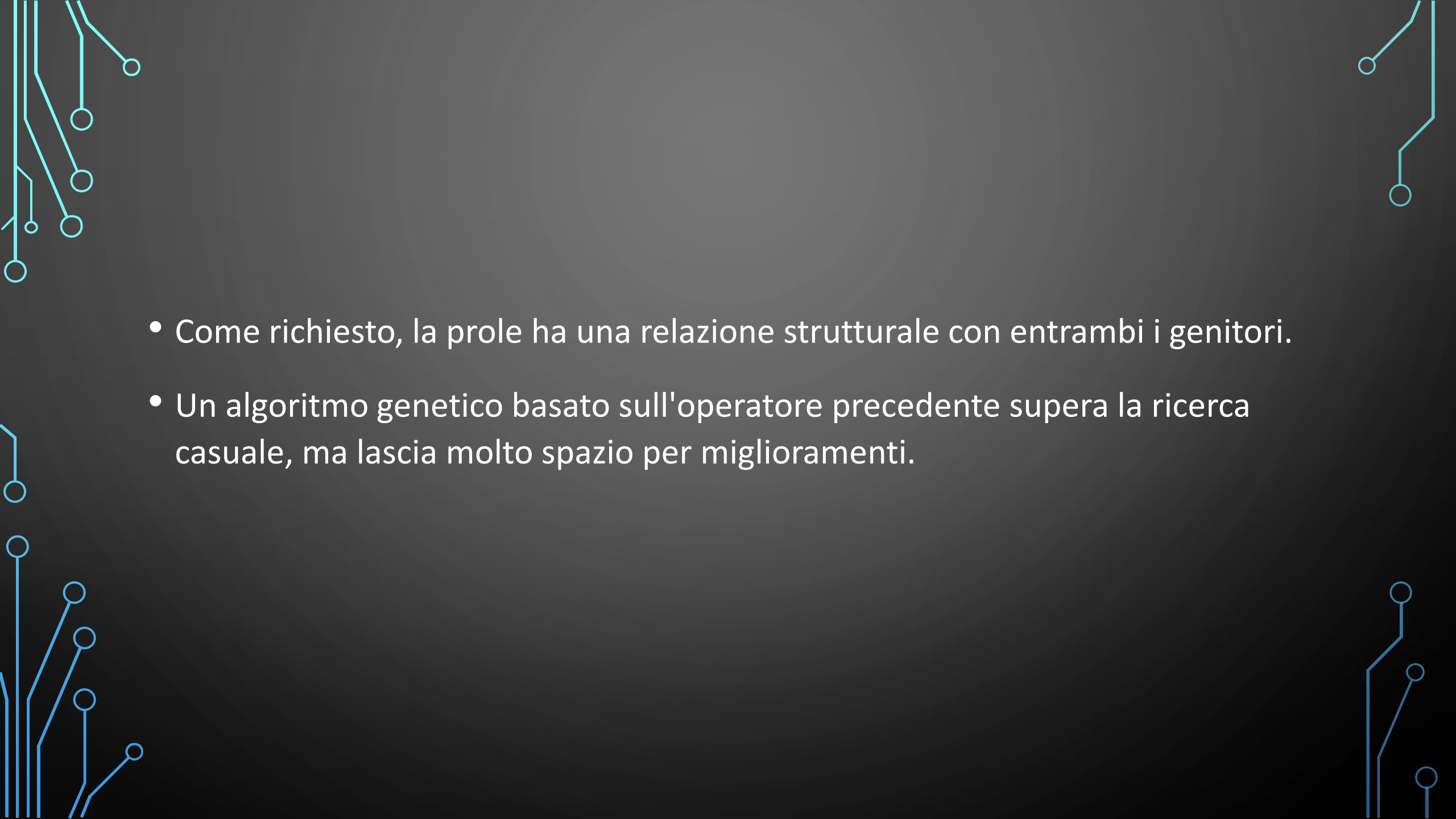
⟨1 2 3 4 5 6 7 8 9 10 11 12⟩ e **⟨7 3 1 11 4 12 5 2 10 9 6 8⟩**

e la parte scelta è

(4 5 6 7),

la prole risultante è

⟨1 11 12 4 5 6 7 2 10 9 8 3⟩

- 
- The background is a dark gray gradient. In the corners, there are decorative elements resembling circuit board traces or neural network connections. These consist of thin, light blue lines that branch out and terminate in small circles. The lines are more dense in the top-left and bottom-left corners, and more sparse in the top-right and bottom-right corners.
- Come richiesto, la prole ha una relazione strutturale con entrambi i genitori.
 - Un algoritmo genetico basato sull'operatore precedente supera la ricerca casuale, ma lascia molto spazio per miglioramenti.



GRAZIE PER L'ATTENZIONE!