Reinforcement Learning per un Genome Assembler de novo

Strumenti formali per la Bioinformatica
A. A. 2023/2024

Riccardo Napoli

Contenuti

Obiettivo

Questo studio mira a migliorare il modello di Reinforcement Learning (RL) combinato ad un algoritmo genetico per l'assemblaggio genomico di Padovani et Al. **[1].**

Obiettivo: includere il RL **end-to-end**, anche nelle operazioni che svolge l'algoritmo genetico.

[1] Kleber Padovani et al. "A step toward a reinforcement learning de novo genome assembler". In: (2024). doi: https://doi.org/10.48550/arXiv.2102.02649

Il problema

Ricostruire un genoma a partire da porzioni più piccole chiamate **reads.**

Reads: sequenze di nucleotidi (A, C, T, G) per DNA ed RNA o sequenze di amminoacidi nel caso di proteine.

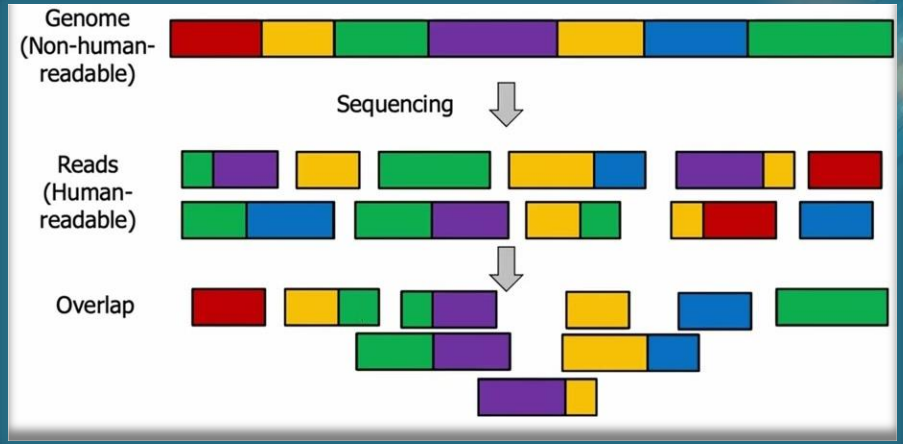
Gli attuali strumenti, i sequenziatori, non ci consentono di "leggere" genomi interamente per via della loro lunghezza.

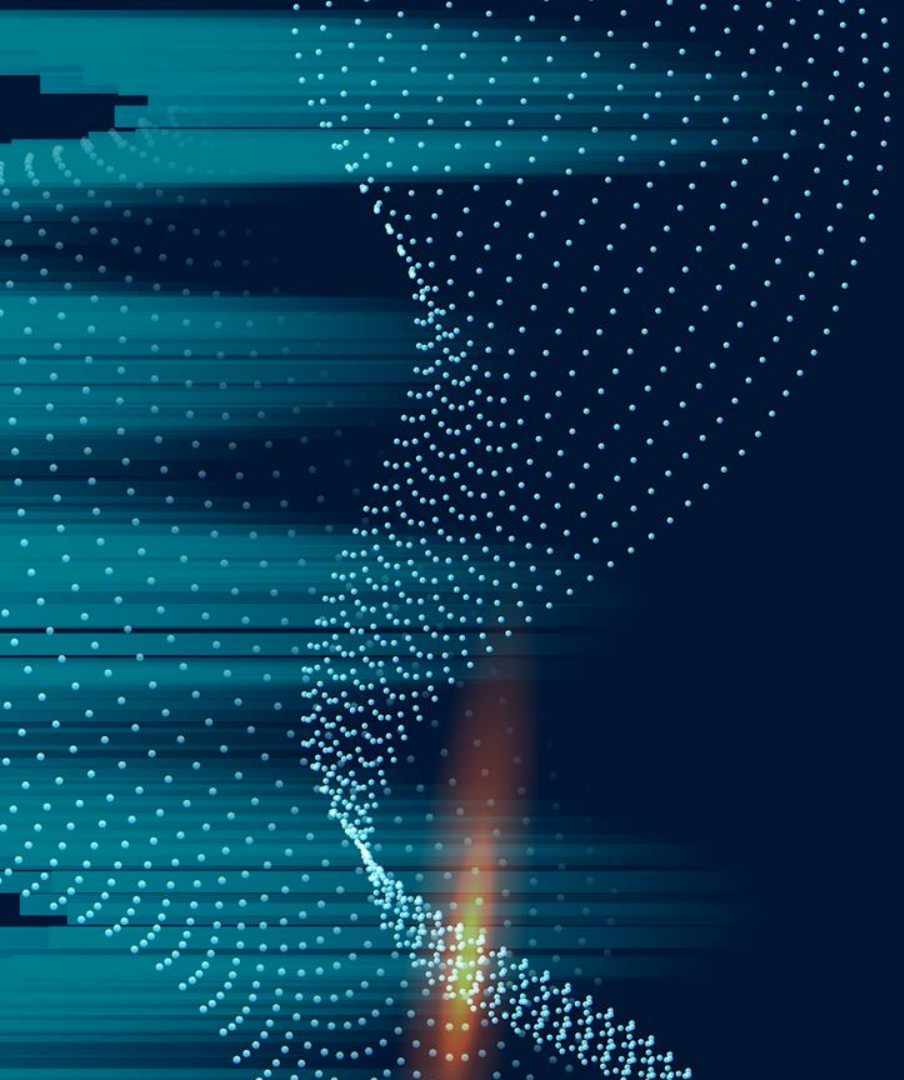
Shotgun Sequencing

Il genoma viene diviso randomicamente in segmenti più piccoli sotto forma di sequenze di testo;

L'assemblatore ricongiungerà i frammenti nel corretto ordine analizzando il loro overlap.

Frequenti errori come l'aggiunta, la rimozione o una lettura sbagliata di un nucleotide.





Assembling de novo

Ricostruzione di un genoma a partire dalle reads, senza il genoma originale.

Importante: pochi genomi presenti in natura si conoscono interamente.

Rientra nella classe dei problemi computazionali più complessi, i problemi NP-difficili.

Quale strategia adottare?

Overlap-Layout-Consensus

- Reads come nodi in un grafo di overlap;
- Archi: punteggio di overlap tra le reads;
- Genoma originale: è il cammino che collega tutte le read con lo score di overlap più alto;
- Corrisponde al cammino Hamiltoniano, problema NP-Difficile.

Grafo di De Bruijn

- Reads divise in sottosequenze di lunghezza fissa, i k-mer;
- Nodi: (k-1)-mers, Archi: k-mers;
- Corrisponde al cammino Euleriano, problema NP-Difficile.

Overlap

L'**overlap** tra due stringhe *s* e *s'* coincide con la lunghezza della più lunga sottostringa presente alla fine di *s* e all'inizio di *s'* o viceversa.

Reinforcement Learning

Differente dai problemi di training supervised e unsupervised: non vi è bisogno nè di dati con label nè di un training set.

Alleniamo il modello a compiere delle scelte in base ad un ambiente in cui si trova attraverso **ricompense e penalità**, con l'intento di raggiungere un obiettivo.

Reinforcement Learning

Il modello ottiene la reward complessiva ottimale a partire da un Processo decisionale di Markov.
Definiamo:

AZIONI

Le decisioni che portano da uno stato all'altro dell'apprendimento

STATI

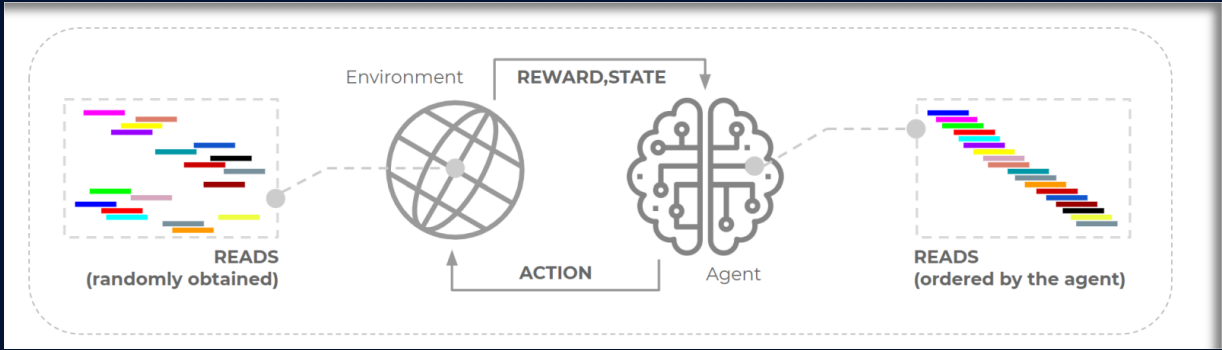
Viste dell'ambiente di training

POLICY

Una serie di regole per attuare una strategia in base all'ambiente in cui si trova l'agente.

Genome assembly con RL

L’agente, scegliendo la read da aggiungere alla sequenza, compie delle azioni, con il rispettivo punteggio di reward. Ripetendo questo processo iterativamente, sarà in grado di trovare la permutazione di reads originale.



Q-Learning

L’algoritmo utilizzato usa una sottocategoria del RL, il **Q-learning**:

- Non viene definita specificatamente una policy;
- L'agente stesso apprende iterativamente la strategia che produce il reward ottimo;
- Pre-computando il reward futuro, l'algoritmo riesce in maniera più efficace a predire l’azione da intraprendere.

Q-Table

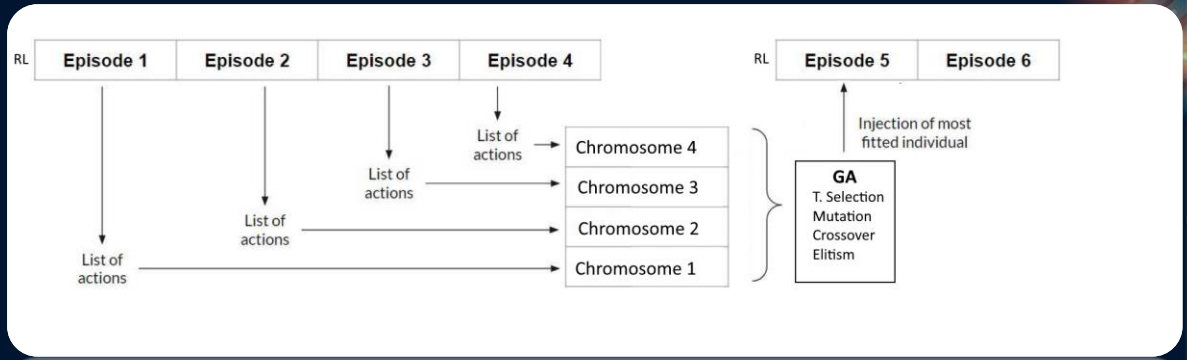
Una particolare struttura, la **Q-table**, viene aggiornata considerando il q-value dell'iterazione precedente, un tasso di apprendimento arbitrario e il reward.

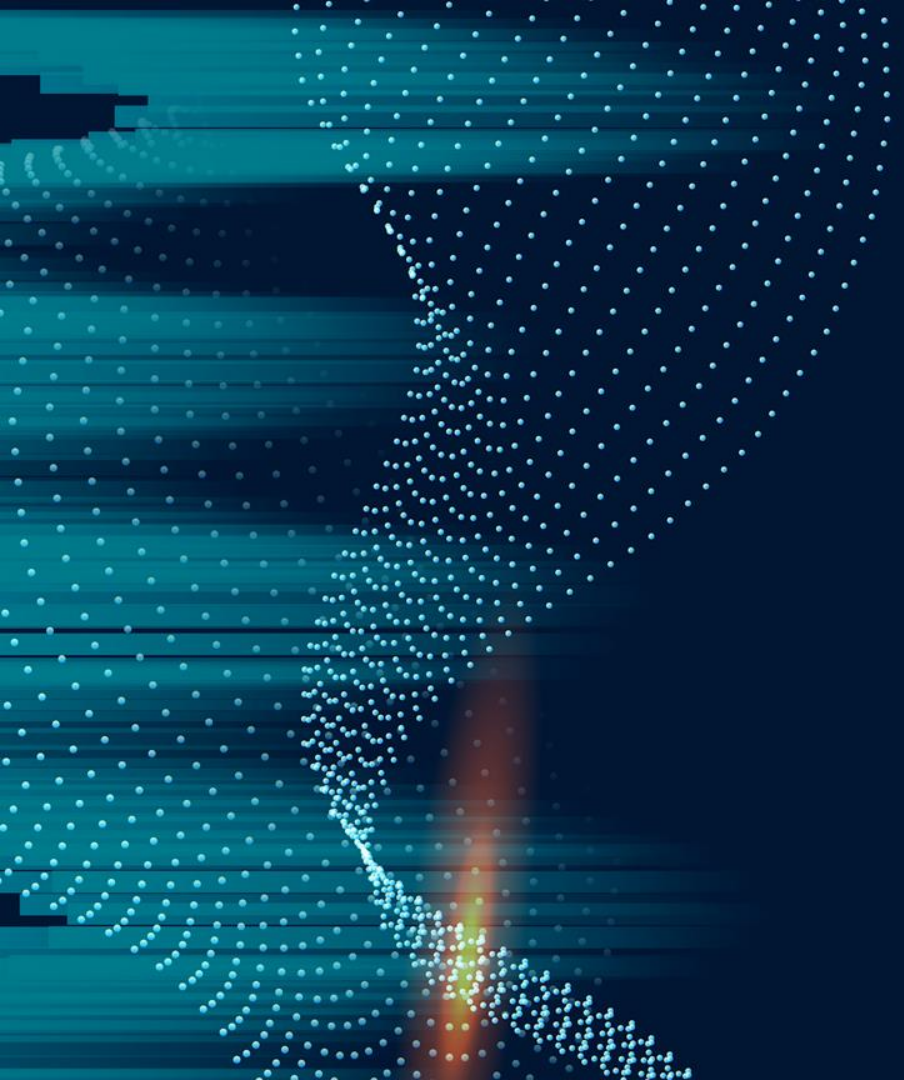
Per l'assemblaggio di un genoma, la q-table viene aggiornata analizzando l'overlap massimo che le successive read fornirebbero.

Gymnome Assembly

L’algoritmo si divide in:

- Scelta di azioni con ε-**greedy policy**;
- Terminazione di un episodio con aggiunta del cromosoma in un pool;
- Esecuzione dell'**algoritmo genetico** sul pool per “evolvere” la popolazione.





ε-Greedy Policy

ε indica la casualità nell'esplorazione: ad ogni step l'agente esplora le reads in maniera aleatoria con probabilità ε, o greedy con probabilità $1-\varepsilon$.

Il valore di ε diminuisce gradualmente.

Per i primi episodi l'esplorazione è casuale, col passare degli episodi si tenderà sempre più ad utilizzare la tecnica greedy.

Stati assorbenti

L’esplorazione viene reiterata fino ad entrare in uno stato «**assorbente**».

Gli stati che richiedono esattamente tutte le reads per essere raggiunti sono assorbenti.

Si entra in uno stato assorbente anche quando la read presa non ha overlap con la precedente.

Una volta raggiunto uno stato assorbente l'episodio termina.

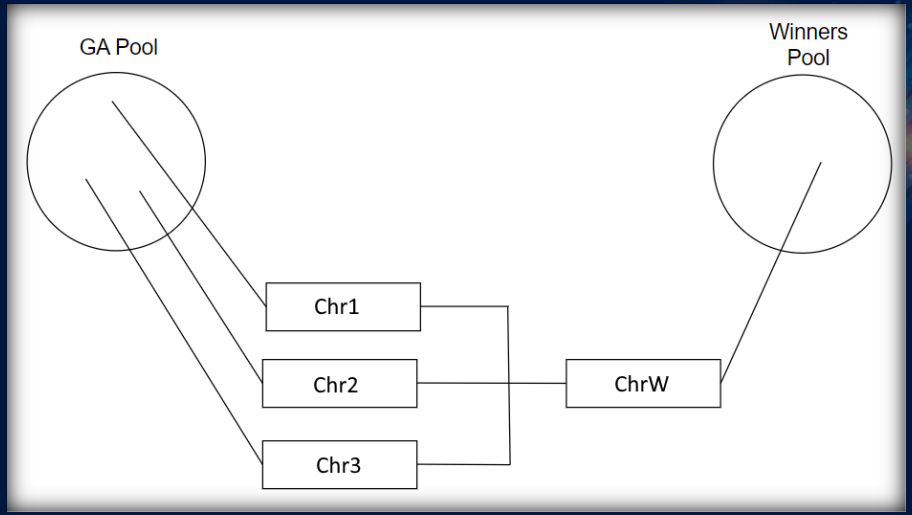
Algoritmo Genetico

Si divide nelle seguenti quattro fasi, per 300 generazioni:

- **Selezione da torneo;**
- **Mutazione;**
- **Crossing-over;**
- **Elitismo.**

Selezione da torneo

- Si estraggono a caso *n=3* cromosomi dal pool;
- Si calcola il punteggio di overlap di tutte le read che compongono i cromosomi;
- Si inserisce in un pool di vincitori di stessa dimensione il cromosoma con il punteggio maggiore.

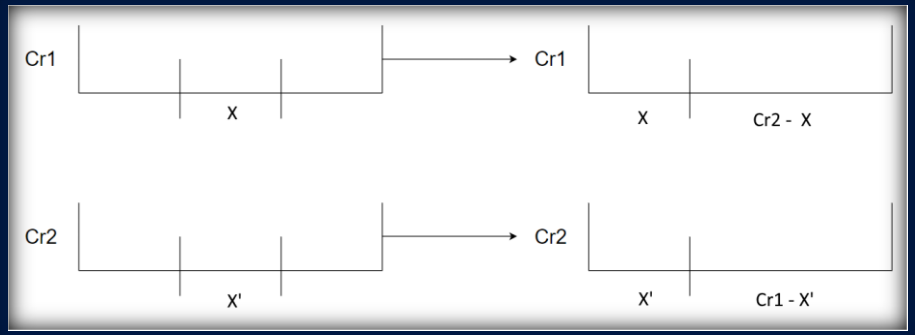


Il pool di vincitori verrà utilizzato per il resto delle operazioni.

Mutazione

Nella fase di mutazione, nel 20% dei casi, due read nei cromosomi del pool di vincitori vengono scambiate e sostituite ai cromosomi del pool originale, per evolvere la popolazione.

Crossing-over



Dati due cromosomi *(i, j)* del pool dei vincitori, il 70% delle volte vengono combinati e sostituiti ai cromosomi *(i, j)* della popolazione originale.

Viene selezionato un intervallo casuale di read *X*.

Da un cromosoma vengono aggiunte le reads appartenenti all'altro, meno quelle che sono già presenti nell'intervallo, per escludere reads doppie.

Elitismo

Il punteggio di overlap di ogni cromosoma del pool viene ricalcolato.
Se il cromosoma di punteggio massimo ha uno score di overlap maggiore dei precedenti viene aggiornato il valore massimo, altrimenti viene iniettato nel primo slot della popolazione originale.

Concluso l'algoritmo genetico ed ottenuto il cromosoma con punteggio di overlap massimo tra le generazioni è possibile evolvere la popolazione.
L' esplorazione riparte da quel cromosoma, che viene scelto interamente dall'algoritmo.

Modifiche proposte

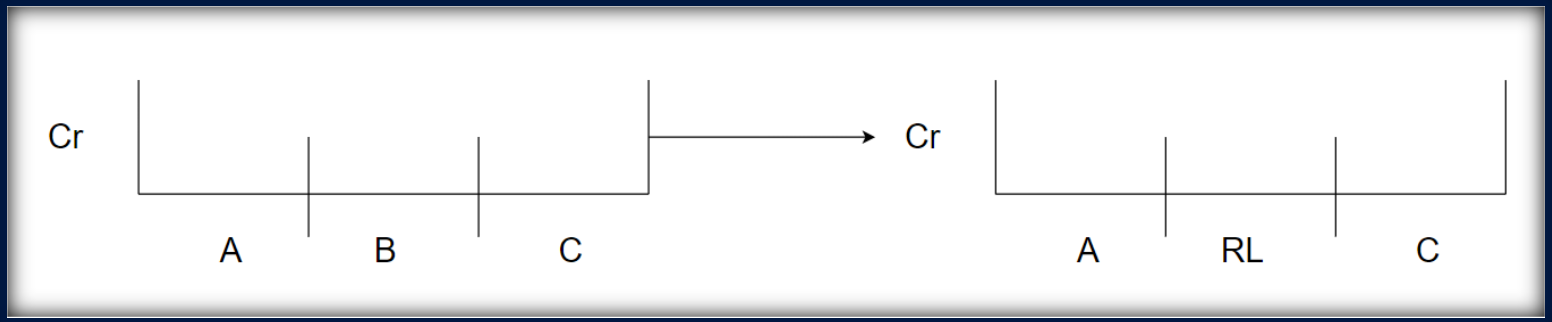
L'algoritmo genetico per l'evoluzione del pool è puramente iterativo e probabilistico.

Le modifiche proposte mirano all'implementazione di un assembler de novo con **RL end-to-end**, anche nella **mutation** e nel **crossing-over**.

La discussione dei risultati si concentrerà solo sul primo dei genomi di riferimento.

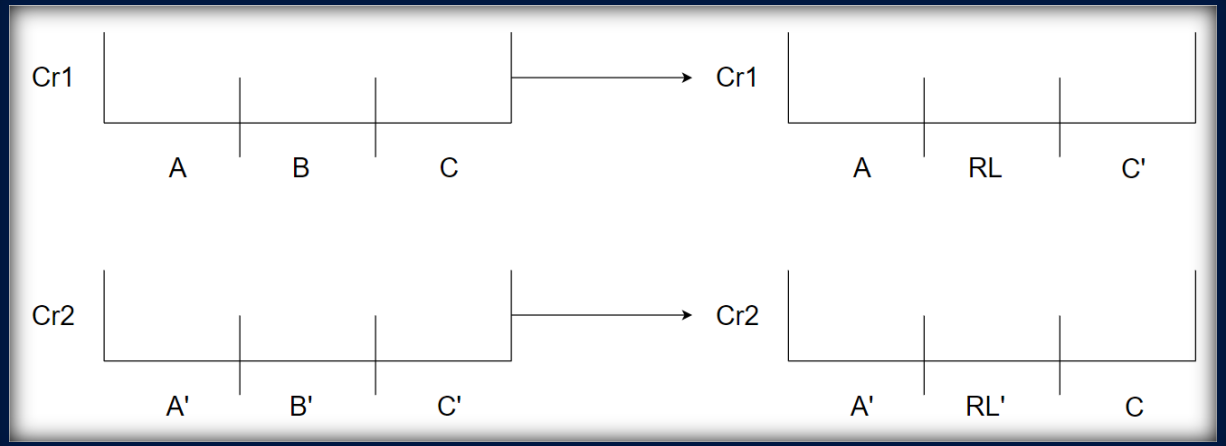
Nuova mutation

L'operazione di **mutation** proposta prevede la divisione del cromosoma in tre parti e il riutilizzo del q-learning sulla parte centrale del cromosoma.



Nuovo crossing-over

Dati i cromosomi *Cr1* e *Cr2*, il RL per la parte centrale di *Cr1* viene eseguito con le reads che compongono la parte centrale di *Cr2* e viceversa,
per cui è possibile effettuare il crossing-over solo nel caso in cui lo scambio **non** formi cromosomi con reads ripetute.



Metrica e Risultati

Per la valutazione dei risultati si è considerato inizialmente solo il primo genoma, di complessità più semplice, per poi fare un test su tutti i genomi.

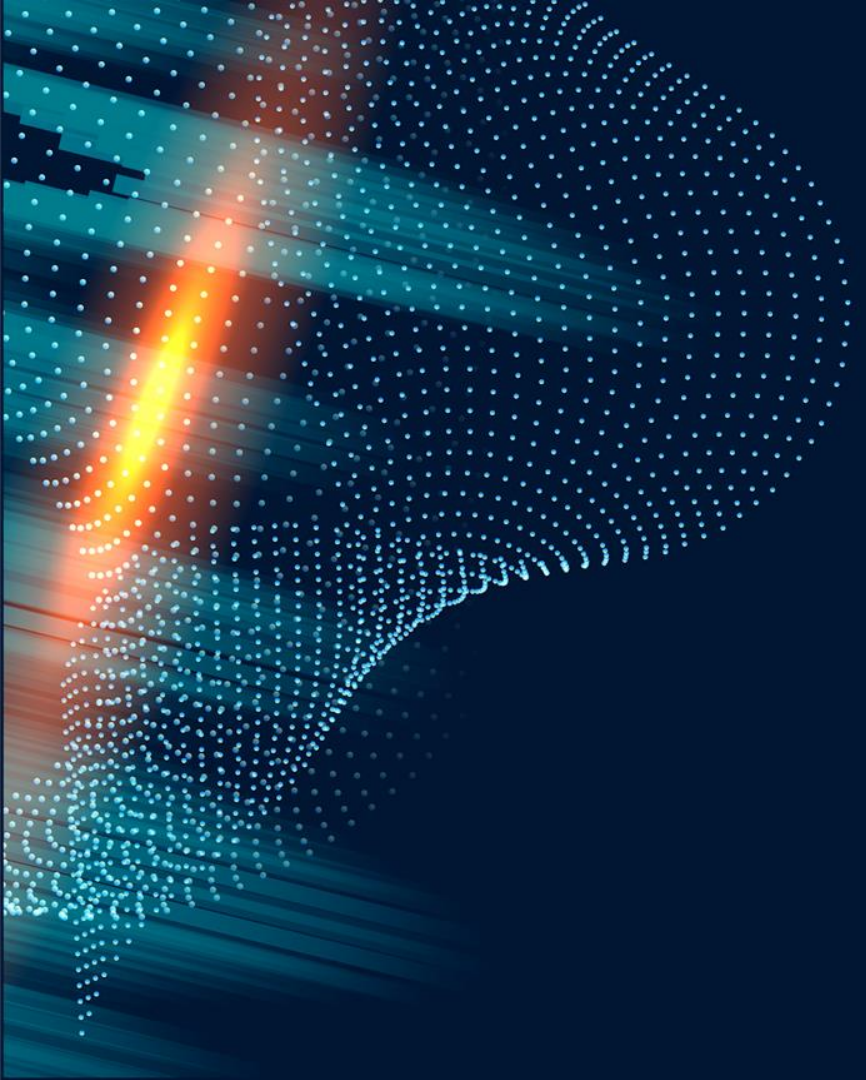
Reward Measure (RM): un episodio è avvenuto con successo se il valore della total reward ottenuto è **maggiore o uguale** di quello prodotto dal caso in cui l'agente esplori la permutazione ottimale delle reads.

Risultati

Caratteristiche del sistema: Processore: 13th Gen Intel(R) Core(TM) i7-13700H 2.40 GHz RAM installata: 8,0 GB Sistema operativo a 64 bit, processore basato su x64					
Algoritmo Originale			Algoritmo Proposto		
Episodi	Tempo (min)	RM Media	Episodi	Tempo (min)	RM Media
1000	0.5	98.70%	1000	1	90%
10000	3	99.85%	10000	15	88.05%
20000	5	99,95%	20000	28	99.08%
30000	7	99.96%	30000	50	82.93%

Risultati

Algoritmo Originale		Algoritmo Proposto	
Tempo Totale	RM Media	Tempo Totale	RM Media
17h 03m	80.87%	174h 24m	67.48%



Risultati

Le prestazioni dell'algoritmo proposto reggono il confronto con l'algoritmo originale in termini di RM, in particolare per il caso del genoma più semplice.

Tuttavia e 4 chiamate al q-learning per ogni elemento del pool, unite ai controlli necessari negli algoritmi proposti, rallentano sensibilmente la computazione.

Possibili Sviluppi

- Crossing-over con RL dinamico, implementato alla maniera originale;
- Ulteriore ottimizzazione e parallelizzazione del codice;
- Scelta **variabile** del numero di componenti del pool, di episodi di q-learning interno e di generazioni dell'algoritmo genetico, in base alla dimensione del genoma.

Grazie per l’attenzione!