Uno studio preliminare per la fattorizzazione di Lyndon per l'allineamento multiplo con gli algoritmi genetici

Mihail Purice[1] and Maria Lombardi[2]

[1]Università degli Studi di Salerno, Fisciano (SA)

July 30, 2024

Abstract

La bioinformatica ha visto una rapida crescita grazie ai progressi nei metodi di sequenziamento e assemblaggio del genoma. In particolare, l'assemblaggio del genoma rimane una sfida computazionale molto complessa. Questo lavoro esplora l'applicazione di tecniche di machine learning, come gli algoritmi genetici e la fattorizzazione di Lyndon, per migliorare l'assemblaggio del genoma. L'approccio descritto utilizza una combinazione di analisi delle sequenze, identificazione dei marcatori e applicazione della fattorizzazione di Lyndon diretta (CFL) per ottimizzare la ricostruzione delle sequenze di DNA. Le letture del DNA sono decodificate utilizzando la fattorizzazione di Lyndon, facilitando il calcolo degli overlap basato sui numeri derivati dalle codifiche, e successivamente vengono elaborate attraverso un algoritmo genetico per affinare le soluzioni.

1 Introduzione

L'assemblaggio del genoma é uno dei compiti più complessi in genomica, mira a ricostruire un genoma utilizzando software che analizzano piccoli frammenti di DNA, dette "letture" e vengono prodotte da un sequenziatore. Ogni nucleotide del DNA è rappresentato dalle lettere 'A', 'C', 'G' e 'T' che sono: adenina, citosina, guanina e timina. Esistono due principali strategie di assemblaggio: l'approccio de novo e l'approccio comparativo. L'approccio comparativo è più semplice e richiede un genoma di riferimento per confrontare le "letture", mentre l'approccio de novo può assemblare nuovi genomi senza un riferimento preesistente. Questo è importante, dato che molti genomi di microrganismi sono ancora sconosciuti. Tuttavia, l'assemblaggio de novo è un problema computazionale complesso, classificato come NP-hard. Nonostante l'assemblaggio del genoma rimanga un problema aperto, alcune metodologie di machine learning sono state esplorate per migliorare le soluzioni disponibili. L'aumento della potenza di calcolo e delle capacità di archiviazione ha incentivato l'applicazione del machine learning, con risultati promettenti. Questo progresso ha inoltre riacceso l'interesse per l'uso dell'apprendimento per rinforzo (RL) e gli Algoritmi Genetici (GA).

2 Approccio Utilizzato

L'analisi delle sequenze è essenziale per comprendere la struttura e le relazioni nei dati. La fattorizzazione delle sequenze scompone una sequenza complessa in sottostringhe, facilitando l'identificazione di pattern nascosti. La fattorizzazione di Lyndon fornisce una scomposizione ordinata basata su proprietà lessicografiche, mentre la fattorizzazione di Lyndon diretta (CFL) genera una rappresentazione compatta e unica delle sequenze. Queste tecniche migliorano la manipolazione e lo studio dettagliato delle sequenze. Gli algoritmi genetici, ispirati all'evoluzione naturale, ottimizzano problemi complessi attraverso selezione, crossover e mutazione. Utilizzati in vari campi, dalla biologia computazionale all'ingegneria, sono fondamentali per trovare soluzioni ottimali in spazi di soluzioni complessi.
Nelle sezioni seguenti, esploreremo in dettaglio la fattorizzazione delle sequenze, e la fattorizzazione di Lyndon.

2.1 Fattorizzazione di Lyndon

La fattorizzazione di Lyndon è un metodo di scomposizione particolarmente significativo, in quanto garantisce proprietà uniche e utili per l'analisi delle sequenze. Una parola di Lyndon è definita come una sequenza di caratteri che è lessicograficamente più piccola di ciascuno dei suoi suffissi non vuoti. Questo tipo di fattorizzazione permette di ottenere una rappresentazione ordinata e standardizzata delle sequenze, facilitando la loro manipolazione.

2.1.1 Fattorizzazione di Lyndon Diretta (CFL)

La Fattorizzazione di Lyndon diretta di una sequenza genera una sequenza di parole di Lyndon $\langle f_1, f_2, \ldots, f_n \rangle$, dove ogni f_i è una parola di Lyndon. Questo processo è essenziale per scomporre una sequenza in elementi fondamentali che conservano le proprietà di Lyndon, fornendo una rappresentazione compatta e unica della sequenza.

2.2 Algoritmi Genetici

Gli algoritmi genetici (GA) rappresentano una classe di algoritmi di ottimizzazione e ricerca ispirati ai processi di selezione naturale e genetica. Introdotti da John Holland negli anni '70, questi algoritmi simulano l'evoluzione delle specie per trovare soluzioni ottimali o quasi ottimali a problemi complessi. Gli algoritmi genetici (GA) utilizzano popolazioni di individui, rappresentati da cromosomi, che evolvono nel tempo attraverso operazioni di selezione, crossover e mutazione 1 . Questi operatori permettono la combinazione e la variazione delle soluzioni esistenti per esplorare lo spazio delle soluzioni, vediamoli più nel dettaglio.

2.2.1 Selezione

Nel contesto degli Algoritmi Genetici (GA), l'operatore di selezione è cruciale per determinare quali individui (soluzioni candidate) passano alla fase successiva del processo evolutivo. La tecnica di selezione utilizzata è basata sull'ordinamento decrescente per fitness. Questo processo assicura che le soluzioni più promettenti siano considerate per prime. Questi individui "migliori" vengono poi utilizzati per il crossover e la mutazione, processi che generano nuove soluzioni combinando e modificando le soluzioni esistenti. Selezionando i migliori individui, si aumenta la probabilità di creare soluzioni migliori nelle generazioni successive.

2.2.2 Crossover

Il crossover è un'operazione genetica fondamentale negli algoritmi genetici. Il suo obiettivo è combinare le informazioni genetiche di due genitori (rappresentati come individui) per generare nuovi figli. Questo processo di mescolamento delle caratteristiche dei genitori ha lo scopo di produrre soluzioni potenzialmente migliori. La selezione elitaria è un approccio di selezione utilizzato negli algoritmi genetici. In questo approccio, una percentuale degli individui con la fitness più alta viene automaticamente portata avanti alla generazione successiva. Questi individui "elitari" vengono poi utilizzati come genitori nelle operazioni di crossover. Questo processo aumenta la probabilità di generare soluzioni di alta qualità nelle generazioni successive. Il two-point crossover è un approccio specifico di crossover utilizzato negli algoritmi genetici. In questo metodo, vengono scelti due punti di taglio lungo la lunghezza del genoma dei genitori. Le sezioni del genoma tra questi due punti vengono poi scambiate tra i genitori per creare due nuovi "figli". Questo processo permette di combinare le informazioni genetiche dei genitori in modi nuovi e unici, potenzialmente portando a soluzioni migliori.

2.2.3 Mutazione

La mutazione è un processo che introduce variazioni casuali nei geni degli individui. Queste variazioni permettono di esplorare regioni dello spazio di ricerca che altrimenti sarebbero inaccessibili attraverso la selezione e l'incrocio. Una delle tecniche di mutazione comunemente utilizzate negli algoritmi genetici è la Swap Mutation. In questa tecnica, due posizioni vengono selezionate casualmente all'interno di un individuo e i loro valori vengono scambiati. Questo processo può portare a nuove soluzioni potenzialmente migliori.

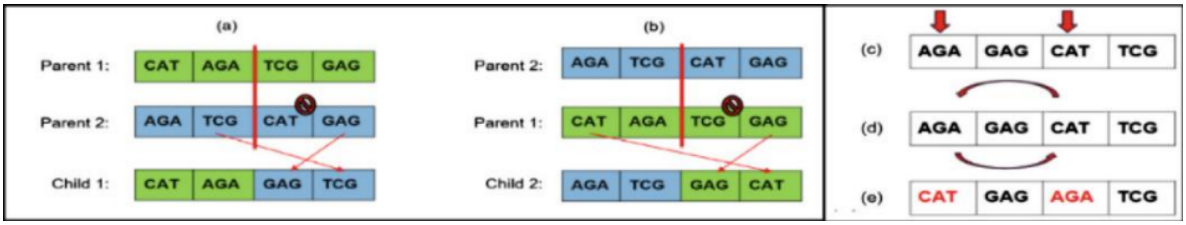

Figure 1: (a) Processo di crossover. I genitori 1 e 2 hanno dimensioni diverse. Nel primo genitore viene selezionato un punto di crossover, quindi è possibile creare il primo figlio. (b) Viene selezionato un punto nel secondo genitore in modo che possa essere creato il secondo figlio. La prole generata non può ospitare letture ripetute. (c) Mutazione cromosomica. Due geni di un cromosoma vengono scelti a caso. (d) Questi geni vengono scambiati. (e) Viene generato un nuovo cromosoma.

3 Metodologia utilizzata

Abbiamo suddiviso il nostro lavoro in tre fasi principali:

- Analisi del codice
- Modifiche realizzate
- Fase di testing

Per l'analisi del codice, abbiamo condotto uno studio approfondito per comprendere la struttura e la logica esistente. Il nostro punto di partenza è stato esaminare il paper di Padovani et al. [1] focalizzandoci sull'approccio 3.2, che utilizza gli algoritmi genetici. Basandoci su questo lavoro, ci siamo posti l'obiettivo di migliorare la tecnica implementando alcune modifiche come l'utilizzo della Fattorizzazione di Lyndon (CFL) per la decodifica delle stringhe.
Nella fase di modifica, ci siamo concentrati sul calcolo degli overlap tra le sequenze. Anziché confrontare i suffissi e i prefissi basandoci sui caratteri, abbiamo adottato la CFL. Questo metodo suddivide la lettura in una o più decodifiche, consentendo il calcolo degli overlap basato sui numeri derivati dalle codifiche.
Durante la fase di testing, i risultati iniziali non erano soddisfacenti. Per risolvere questo problema, abbiamo introdotto una fase di pre-processing delle letture, aggiungendo dei marcatori prima di applicare la CFL.

3.1 Utilizzo e Creazione del Dataset

Siamo partiti dal utilizzare il dataset del paper preso in analisi[1] e successivamente abbiamo generato un dataset di 2000 caratteri scelti casualmente tra le lettere **'A', 'C', 'G', e 'T'**. In seguito, abbiamo suddiviso questo dataset in **N** letture di 100 caratteri ciascuna, con sovrapposizioni tra di loro per coprire l'intero dataset iniziale. Questo processo ci ha permesso di creare il nostro primo individuo, e creare anche il genoma iniziale da confrontare alla fine del processo.

3.2 Costruzione e utilizzo dei marcatori

Dopo aver ottenuto le letture, abbiamo identificato i marcatori presenti in esse. Inizialmente, abbiamo lavorato con un solo marcatore alla volta, verificando la frequenza delle sue ripetizioni nelle letture per applicare correttamente la CFL. Successivamente, abbiamo utilizzato una combinazione di marcatori, con lunghezze diverse, per ottenere una suddivisione più precisa.

3.3 Applicazione della Fattorizzazione di Lyndon (CFL).

Una volta identificati i marcatori, abbiamo suddiviso la nostra lettura applicando la CFL ogni volta che un marcatore era presente in essa, ottenendo così una decodifica in più punti della sequenza, come mostrato nella figura sottostante.

Figure 2: Rappresentazione CFL di una lettura

Questa metodologia ci ha permesso di ottenere risultati più precisi e di migliorare la decodifica delle stringhe rispetto all'approccio originale. Grazie all'uso della CFL e alla fase di pre-processing con i marcatori, siamo riusciti a superare le difficoltà iniziali e a incrementare l'efficacia del nostro algoritmo.

3.4 Utilizzo dell' Algoritmo Genetico

Partendo da questo individuo iniziale, abbiamo generato N individui senza ripetizioni per utilizzare correttamente l'algoritmo genetico mostrato in figura sottostante. Una volta creati questi individui, abbiamo applicato l'algoritmo genetico, che sfrutta due principali operatori: la mutazione e il crossover.

- La **Mutazione** introduce variazioni casuali nelle popolazioni per mantenere la diversità genetica e prevenire il problema del minimo locale.
- Il **Crossover** combina parti di due popolazioni (genitori) per creare una nuova popolazione (figli), cercando di ereditare le migliori caratteristiche da entrambi i genitori.

```
    def run_ga(self, env, population,iterazioni):
        pop_fitness = self._evaluatePopulation(population)

        for generation in range(iterazioni):
            population = sorted(pop_fitness, key=lambda x: x[0],
                reverse=True)

            best_ind = population[0]

            if len(population) < 2:
                return best_ind
            else:
                lenPop = len(population) // 2
                population = population[:lenPop]

            # Crossover
            for i in range(0, len(population) - 1, 2):
                if np.random.rand() < self.crossover_prob:
                    if i < len(population) - 1:
                        child1, child2 = self._crossover(population[i
                            ], population[i + 1])
                        population.append(child1)
                        population.append(child2)


```

```
24
25              # Mutation
26              for i in range(len(population)):
27                  if np.random.rand() < self.mutation_prob:
28                      temp = self._mutation(population[i])
29                      population[i] = temp
30
31              if best_ind not in population:
32                  population.append(copy.deepcopy(best_ind))
33
34              pop_fitness = self._evaluatePopulation(population, gen)
35
36          return best_ind
```

Dopo aver applicato questi operatori a tutti gli individui della nostra popolazione, abbiamo calcolato la fitness di ciascun individuo. Nel nostro contesto, la fitness rappresenta l'overlap sequenziale dei numeri, cioè la misura in cui le sequenze ricostruite corrispondono alle letture originali.
L'individuo con il valore di fitness più alto è stato selezionato per proseguire nelle iterazioni successive. Gli altri $N - 1$ individui sono state modificati nuovamente attraverso gli operatori di mutazione e crossover. Questo processo di selezione e modifica è stato ripetuto per N iterazioni, permettendo all'algoritmo di affinare progressivamente le soluzioni.
Una volta individuato l'individuo migliore, abbiamo proceduto con la ricostruzione del genoma completo. Per valutare la precisione della ricostruzione, abbiamo calcolato la distanza di Levenshtein tra il genoma ricostruito e il genoma iniziale. La distanza di Levenshtein rappresenta il numero di modifiche, inserimenti o eliminazioni necessarie per trasformare una stringa nell'altra, fornendo una misura della similarità tra i due genomi.

4 Ottimizzazione e Valutazione dei Risultati

Durante la fase di test, abbiamo esplorato diverse strategie di ottimizzazione per migliorare le prestazioni del nostro approccio di Fattorizzazione di Lyndon (CFL). Una strategia significativa ha coinvolto l'utilizzo di una combinazione di marcatori. Sfruttando contemporaneamente più marcatori, miravamo a migliorare l'accuratezza e l'efficienza nella ricostruzione delle sequenze. In particolare, abbiamo sperimentato con otto marcatori, ciascuno con una lunghezza superiore a sette. Questi marcatori più lunghi hanno costantemente superato altre combinazioni testate. Per ottenere una ricostruzione accurata delle sequenze, abbiamo dovuto calcolare gli overlap tra i valori interi associati alle decodifiche ottenuto tramite la Fattorizzazione di Lyndon (CFL). L'approccio CFL ci ha permesso di suddividere il dataset in segmenti più piccoli utilizzando le parole di Lyndon. In questo modo, abbiamo garantito l'identificazione corretta delle regioni sovrapposte, contribuendo a risultati più affidabili. Per valutare quantitativamente la qualità dei nostri genomi ricostruiti, abbiamo utilizzato la distanza di Levenshtein. Questa metrica misura la similarità tra due stringhe conteggiando il numero minimo di modifiche di un singolo carattere (inserimenti, cancellazioni o sostituzioni) necessarie per trasformare una stringa nell'altra. Nel nostro caso, abbiamo confrontato il genoma ricostruito dal nostro algoritmo genetico (GA) con il genoma originale. La distanza di Levenshtein risultante, pari a 818, indica la dissimilarità tra le due sequenze.

References

[1] K. Padovani, R. Xavier, A. Carvalho, A. Reali, A. Chateau, and R. Alves, "A step towards a reinforcement learning de novo genome assembler," *arXiv preprint arXiv:2102.02649*, 2021.