Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Simulated Annealing per l'inferenza di mutazioni ricorrenti in alberi tumorali

Relatore: *Prof. Gianluca Della Vedova*
Correlatore: *Dott. Simone Ciccolella*

Relazione della prova finale di: *Stefano Giacomini*
Matricola 829854

Anno Accademico 2020-2021

Indice

Capitolo 1

Introduzione

Recenti sviluppi riguardanti la terapia mirata per la cura di tumori fanno affidamento su un'accurata inferenza della mutazione clonale e della progressione della malattia. Diversi studi sottolineano come capire l'ordine di accumulazione e la prevalenza delle mutazioni somatiche durante la progressione del tumore può aiutare a progettare meglio queste strategie di trattamento.
La maggior parte delle tecniche correntemente utilizzate per l'inferenza di progressioni tumorali si basa su dati provenienti da esperimenti di sequenziamento di massa, dove solo una parte delle delle mutazioni osservabili da un grande numero di cellule è ottenuta, senza la distinzione di quali cellule le contengono. Negli ultimi anni sono stati sviluppati molti approcci computazionali per l'analisi di dati di sequenziamenti di massa con lo scopo di inferire la scomposizione di subcloni tumorali e ricostruire la filogenesi dei tumori (alberi evolutivi).
Il problema principale di questa tecnica è che un campione ottenuto da un sequenziamento di massa contiene un misto sia di cellule sane che di cellule tumorali, e questa mutazione clonale può essere solo stimata dalla porzione di mutazioni osservate.
Il sequenziamento a singola cellula, o *SCS* (single-cel sequencing), promette di creare la migliore soluzione per capire la causa della progressione del tumore. Tuttavia la mancanza di accuratezza, riflessa nell'alta probabilità di falso positivo e nell'alto drop-out allelico, inerente alla tecnologia usata richiede l'uso di metodi che sono in grado di creare inferenze di progressioni tumorali dai dati prodotti dalle correnti tecniche di *SCS*.
Vari metodi sono stati recentemente sviluppati con questo scopo, molti di questi metodi però si basano sull'*ISA* (Infinite Site Assumption), il quale essenzialmente indica come una mutazione possa essere acquisita al massimo una volta e mai essere persa. Un motivo per cui questo viene fatto è che questa assunzione semplificativa porta ad un modello evolutivo il cui problema è risolvibile da un algoritmo in tempo polinomiale, ovvero il modello di filogenesi perfetta.
Questo metodo è sicuro da usare solo in determinate situazioni, come ad esempio l'evoluzione di una popolazione, dove tende ad essere la norma più che un eccezzione. La progressione di cellule tumorali è però invece una situazione piuttosto estrema, dove le cellule sono sotto attacco del sistema immunitario, l'evoluzione è molto veloce e con un'alta probabilità di mutazione.
Come risultato studi su dati prodotti da SCS stanno cominciando a rilevare fenomeni non spiegabili da un modello di filogenesi perfetta. In questo lavoro utilizziamo quindi un modello Car-(k, r), il quale è un modello più generale rispetto al modello di filogenesi perfetta, per permmettere la perdita e la ricorrenza di mutazioni. Una volto che però ci allontaniamo dal modello ideale di filogenesi

perfetta, privo dalla possibilità di errori, passiamo ad un problema NP-hard.
In più, la maggior parte dei metodi attualmente disponibili assumono che la probabilità di falso negativo sia la stessa per tutte le mutazioni. Se questa assunzione è valida per dati provenienti da campioni di DNA, la probabilità di falso negativo in campioni provenienti da RNA può variare a causa dei differenti livelli di espressione genetica. Visto che il nostro approccio è pensato per dati provenienti sia da DNA che da RNA i parametri passati possono essere settati per adattarsi a dati provenienti da uno piuttosto che dall'altro, permettendo quindi anche una probabilità di falso negativo diversa per ogni mutazione.
Partendo dal metodo SASC, un metodo di ricerca di filogenesi che massimizza la likelihood permettendo perdita di mutazioni incorporando il modello di Dollo, abbiamo quindi sviluppato un metodo più generico.

1.1 Motivazioni

L'utilizzo di un modello più generico, che comprende anche mutazioni ricorrenti, diventa particolarmente importante quanto si deve descrivere l'evoluzione virale.
Facendo un esempio molto attuale, il virus SARS-CoV-2, virus che è causa della malattia comunemente chiamata COVID-19, sono state riscontrate 198 mutazioni ricorrenti, la maggior parte delle quali a livello proteico.
É importante lo studio di mutazioni che sono emerse indipendentemente più volte (omoplasia) in quanto sono molto probabilmente adattamenti del virus all'ospite, in questo caso quinid indicano l'adattamento del virus SARS-CoV-2 al genere umano.
Per descrivere meglio anche questi casi è stato quindi sviluppato il metodo di Simulated Annealing per l'inferenza di mutazioni ricorrenti (o recurrent SASC), il quale permette sia la perdita che la ricorrenza di mutazioni, incorporando un modello Car-(k, r).

Capitolo 2

Definizioni

2.1 Problemi di ricorstruzione di alberi filogenetici

Prima di cominciare a parlare del recurrent SASC vero e proprio è bene deinire i concetti che verranno utilizzati.

Un albero filogenetico è un grafico che mostra visivamente la collocazione temporale della separazione fra le linee evolutive che a partire da una data specie ha portato alla formazione di due o più specie diverse attraverso una serie di biforcazioni. Con filogenesi basata su caratteri, dove con carattere si intende una qualunque caratteristica che si può osservare, si intende quella filogenesi che sfrutta alcuni caratteri specifici, un esempio può essere la presenza o meno della colonna vertebrale, per rappresentare la storia evolutiva di alcune specie. Utilizzando questo concetto e applicandolo, invece che alla separazione di due linee evolutive dovuta ad una serie di mutazioni, all'acquisizione di una cellula di una determinata mutazione, si può rappresentare l'evoluzione di cellule tumorali. Come menzionato precedentemente la ricostruzione della progressione evolutiva di cellule tumorali può essere modellato con la costruzione di una filogenesi incompleta basata sui caratteri, dove ogni carattere rappresenta una mutazione. Consideriamo quindi come input una matrice I_{ij} di dimensioni n x m, con n numero delle cellule e m numero delle mutazioni, e dove $I_{ij} = 0$ indica che la sequenza i non ha la mutazione j, $I_{ij} = 1$ indica la presenza della mutazione j nella sequenza i e ? indica che non ci sono abbastanza informazioni sulla presenza/assenza di una mutazione j nella sequenza i. Questa incertezza sulla presenza di una mutazione deriva da una insufficiente precisione nel sequenziamento del DNA/RNA.
Oltre a questo ci sono altri problemi derivanti dal sequenziamento, infatti i valori della matrice di input possono anche contenere falsi positivi e falsi negativi, mentre la probabilità di falso positivo è in genere abbastanza bassa, la probabilità di falso negativo può essere alta e variare da una mutazioni all'altra. Quindi, detta E_{ij} la matrice n x m di output finale, ovvero la matrice binaria senza errori o rumore prodotto dall'algoritmo, allora α_j indica la probabilità di falso negativo della mutazione j, mentre invece β indica la probabilità di falso positivo.
Di conseguenza, per ogni valore di E_{ij}, valgono le seguenti:

$$P(I_{ij} = 0|E_{ij} = 0) = 1 - \beta P(I_{ij} = 0|E_{ij} = 1) = \alpha_j$$

$$P(I_{ij} = 1|E_{ij} = 0) = \beta P(I_{ij} = 1|E_{ij} = 1) = 1 - \alpha_j$$

Puntiamo quindi a trovare una matrice che massimizzi la likelihood della matrice osservata I sotto la probabilità di falso negativo/positivo e i valori mancanti. Il nostro lavoro contiene in più la perdita di mutazioni e la possibilità di avere più ricorrenze della stessa mutazione, andiamo quindi a definire $P(L(j)) = \gamma_j$ come la probabilità di perdità della mutazione j e il set di variabili c_j per $j = 1, ..., m$ che denota il numero totale di perdite per la mutazione j nell'albero. Definiamo poi $P(D(j)) = \delta_j$ come la probabilità di creare una copia della mutazione j e il set di variabili f_j per $j = 1, ..., m$ che denota il numero totale di copie per la mutazione j presenti nell'albero.
Un albero filogenetico T di un set C di m mutazioni e n cellule (affette da queste mutazioni) è definito come un albero con radice i cui nodi interni sono etichettati con le mutazioni di C, mentre le foglie sono etichettate dalle cellule. É importante far notare come l'etichettamento dei nodi deve rispettare alcune restrizioni a seconda del modello evolutivo che stiamo considerando. Per esempio, in una filogenesi perfetta, non possono esistere due nodi con la stessa etichetta. Questa è un'alternativa della classica filogenesi basata su caratteri, dove l'albero T è definito come un set di caratteri e dove le foglie non hanno nessuna etichetta e rappresentano specie differenti.
Lo *stato* di un nodo x è definito come il set di mutazioni acquisite ma non perse nel percorso dalla radice a x. Lo stato di ogni foglia l di T è rappresentato da un vettore binario di lunghezza m, chiamato profilo del genotipo, che denotiamo come $D(T, l)$, dove $D(T, l)_j = 1$ se e solo se la foglia l ha la mutazione j e 0 altrimenti.
Diciamo che l'albero T *codifica* una matrice E se esiste una mappatura σ delle righe (cellule) di E nelle foglie di T tale che $E_i = D(T, \sigma)_i$ per ogni riga i di E, dove σ_i denota l'immagine di una riga i attraverso la mappatura σ. Detto in maniera meno formale, σ_i è il nodo dell'albero filogenetico corrispondente al nodo a cui la cellula i è attaccato. Notare come la matrice E è completamente caratterizzata dalla coppia $D(T, \sigma)$. Dunque il nostro problema può essere espresso come trovare l'albero T che massimizzi la seguente funzione obiettivo:

$$\max \sum_{j}^{m} [-c_j \log(1 - P(L(j))) - f_j \log(1 - P(D(j))) + \sum_{i}^{n} \log(P(I_{ij}|D(T, \sigma_i)_j))]$$

Facciamo inoltre notare come i valori assegnati alle entrate sconosciute della matrice di input non contano nel calcolo della funzione obiettivo, ovvero $P(I_{ij} = ?|E_{ij} = 1) = P(I_{ij} = ?|E_{ij} = 0)$. Per semplificare il calcolo della likelihood supponiamo che $P(I_{ij} = ?|E_{ij} = 1) = P(I_{ij} = ?|E_{ij} = 0) = 1$. In più σ può essere computato direttamente da T, lasciando quindi T come unica variabile da ottimizzare.

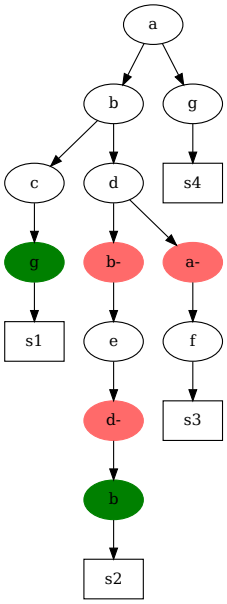
2.2 Dollo-k e Camin-Sokal-k

La regola di parsinomia di Dollo assume che in filogenesi ogni singola mutazione è introdotta una sola volta nella storia evolutiva, ma che invece perdite della stessa possono occorrere un numero infinito di volte. Una restrizione del modello di Dollo è ottenibile limitando il numero di perdide per ogni mutazione. Chiamiamo quindi Dollo-k il modello evolutivo in cui ogni mutazione può essere acquisita solo una volta e persa al massimo k volte. I casi speciali Dollo-0 e Dollo-1 corrispondono rispettivamente ai modelli di perfetta e persistente filogenesi.
Invece il modello di Camin-Sokal permette molteplici occorrenze della stessa mutazione ma nessuna perdita. Definiamo quindi Camin-Sokal-k il modello evolutivo in cui una mutazione può avere al più k ripetizioni e non può mai essere persa. Il caso speciale Camin-Sokal-0 corrisponde al modello di filogenesi perfetta.

Il modello utilizzato in questo lavoro è il Car-(k, r), un modello che va ad unire i modelli introdotti precedentemente e che quindi ammette al massimo k perdite (o backmutation) e r ripetizioni (o recurrent mutation). Dove i modelli di filogenesi perfetta e filogenesi persistente si ottengono rispettivamente con i casi Car-(0, 0) e Car(1, 0). Il problema di ricorstruzione della filogenesi su un modello Car-(k, r) è NP-completo per ogni $k > 1$ e $r > 0$.
Visto che il modello Carl(l, r) utilizza sia backmutation che recurrent mutation introduciamo due nuovi tipi di nodi nell'albero filogenetico, per esprimere le perdite e le ripetizioni delle mutazioni. Per ogni mutazione p creiamo k nuove mutazioni ${p^-}_l$ per ogni $l \in \{i, ..., k\}$, rappresentanti le possibili perdite della mutazione p, in più creiamo r nuove mutazioni p_j per ogni $j \in \{i, ..., r\}$ reppresentanto le possibili copie della mutazione p. In più imponiamo delle nuove regole: che ogni nodo etichettato da una perdita di una mutazione p^- debba essere discendente di un nodo etichettato dall'acquisizione di una mutazione p, e che ogni nodo etichettato da p non possa essere discendente di un'altro nodo etichettato con p, a meno che questo non sia prima stato perso con un nodo p^-.
É importante sottolineare come, a differenza del modello a filogenesi perfetta, con l'introduzione di perdite e ricorrenze potremmo avere più alberi che sono soluzione del problema.

2.2.1 Modello utilizzato

Nel modello utilizzato sono state in più aggiunte le seguenti restrizioni: il numero massimo di perdite nell'intero albero è al massimo d e il numero di copie è al massimo z.
Questo fa si che la variabile c sia soggetta a (i) $c_j \geq k \ \forall j$ e (ii) $\sum_j^m c_j \geq d$ e che in maniera analoga la variabile f sia soggetta a (i) $f_j \geq r \ \forall j$ e (ii) $\sum_j^m f_j \geq z$. É inoltre importante notare come con un numero di mutazioni m non piccolo, settare la variabile d con un valore abbastanza piccolo, per esempio 5, equivale a settare $k \leq 1$, rendendo quindi il valore di k irrilevante; un discorso analogo può essere fatto per quanto riguarda r e z.



	a	b	c	d	e	f	g
s_1	1	1	1	0	0	0	1
s_2	1	1	0	0	1	0	0
s_3	0	1	0	1	0	1	0
s_4	1	0	0	0	0	0	1

Figura 2.1: Esempio di una matrice E (n=4, m=7), a destra, che esprime le cellule $\{s_1, ..., s_4\}$ affette dal set $C = \{a, ..., g\}$ di mutazioni. L'albero T, a sinistra, è un albero di filogenesi di cellule tumorali raffigurante la matrice E, nel quale i nodi rappresentanti le backmutation sono colorati di rosso e i quelli rappresentanti le recurrent mutation sono colorati di verde. É importante far notare che la matrice non permette una filogenesi perfetta e che T è una filogenesi Car(1, 1)

2.3 Simulated Annealing

Il fatto che (i) e (ii), e il fatto che vogliamo cercare l'albero che massimizzi la likelihood, rendono il problema di riscostruzione della filogenesi computazionalmente NP-hard sotto il modello Car(k, r) per ogni k o $r > 0$. Per questo motivo si è deciso di utlizzare il metodo del Simulated Annealing (SA) per cercare l'albero che massimizzi la likelihood di una determinata matrice di input, soddisfacendo il modello Car(k, r), dove k e r sono dati come input. SA è una metaeuristica che sfrutta un metodo di ricerca randomico che esplora la regione delle soluzioni ammissibili alla ricerca dell'ottimo, esattamente come tutte le altre metaeuristiche non c'è certezza che SA trovi la soluzione ottima della funzione obiettivo in un numero finito di step.
SA è però in grado, rispetto ad altri metodi deterministici, di evitare di rimanere bloccato in ottimi locali in quanto può accettare anche step non migliorativi; nello specifico sia data X_c, soluzione ammissibile, e Z_c, valore della funzione obiettivo nel punto X_c, calcolo X_n, punto nell'intorno di X_c calcolato da una mossa randomica, se Z_n, valore della funzione obiettivo in X_n, è migliore di Z_c, allora X_n diventa il nuovo X_c, altrimenti ho comunque una probabilità che questo avvenga ed è pari a: $e^{\frac{Z_n - Z_c}{T}}$. Con T che è una variabile chiamata *temperatura* scelta inizialmente e che continua a diminuire nel tempo dopo un certo numero di mosse non migliorative.
Come si può intuire dalla formula, una temperatura alta indica una più alta probabilità di accettare mosse molto peggiorative, favorendo l'*exploration*, ovvero la ricerca all'interno della funzione per evitare di rimanere incastrati in ottimi locali, invece man mano che la temperatura diminuisce

anche la probabilità di accettare mosse molto peggiorative, favorendo quindi l'*exploitation*, ovvero la ricerca dell'ottimo nell'intorno in cui mi trovo, con $T \approx 0$ non accetto più soluzioni peggiorative. Il criterio di arresto dell'algoritmo in genere è l'aver fatto un certo numero di iterazioni con una T minima, scelta inizialmente.

Capitolo 3

Metodo

Si discute ora il metodo col quale si ricostruisce l'albero evolutivo massimizzando la *likelihood*, ovvero il valore che indica la bontà della soluzione trovata. A tal fine sono state fatte diverse aggiunte alla codebase di **SASC** e quindi è bene, come prima cosa, analizzare le strumentazioni, ovvero il linguaggio di programmazione e le librerie principali, usate nel progetto.

3.1 Introduzione agli strumenti utilizzati

3.1.1 Librerie e linguaggio

Nel progetto è stato utlizzato come linguaggio *C*, ai fini del progetto non sono stati aggiunti nuovi file sorgenti ma sono state fatte aggiunte e modifiche all'interno della *codebase*.
Bisogna, quindi, citare una serie di librerie esterne utilizzate in diverse parti del codice C:

- La libreria **vector**, che permette l'implementazione della struttura dati dei *vettori* nel linuaggio *C*
- La libreria **tree**, che permette l'implementazione di alberi evolutivi con nodi etichettati nel linguaggio *C*, rapprentando anche backmutation e recurrent mutation
- La libreria **mt19937ar**, che consente di generare numeri casuali in vari maniere, nello specifico nel progetto viene utilizza per generare numeri casuali all'interno di determinati range o per generare unsigned int a 32-bit casuali

3.1.2 Graphwiz

É un sotware open source per la visualizzazzione di grafi che, data la descrizione di un grafo, ne crea il diagramma in vari formati quali, per esempio pdf o SVG. La descrizione del grafo da creare può essere passata o direttamente nel terminale o attraverso un file con estensione *GV*. I file *GV* contengono descrizioni riguardanti grafi scritte nel lingiaggio *DOT*, il quale è un linguaggio per la rappresentazione di grafi che può essere interpretato da Graphwiz o da altri programmi che lo usano internamente.
Graphwiz possiede vari opzioni per quanto riguarda il colore e la forma dei nodi ma anche font,

stili di archi e in generale altre opzioni per personalizzare l'estetica di un grafo. Il linguaggio DOT è stato utlizzato come output del codice sia per stampare sul terminale gli alberi che per creare un file GV di output contente l'albero migliore.

3.2 Input

Iniziamo quindi a descrivere come funziona il codice di *recurrent SASC* vero e proprio.
La prima parte riguarda ai dati di input che vengono passati, che ruguardano sia il modello utilizzato, Car-(k, r), sia la matrice indicante le mutazioni delle varie cellule e in più anche i dati relativi a falso positivo e falso negativo.
Riguardo al modello Car-(k, r) vengono passati inanzitutto i valori k e r del modello, i quali indicano rispettivamente il numero massimo di backmutation e di recurrent mutation di ongi mutazione e d e z che sono il numero massimo di backmutation e recurrent mutation presenti nell'intero albero, questi ultimi dati se non vengono passati sono impostati +inf, le probabilità γ e δ, che sono le probabilità di aggiungere backmutation e recurrent mutation, le quali possono essere diverse per ogni mutazione (e quindi passate attraverso un file) o le stesse per ogni mutazione dell'albero.
Per quant riguarda invece la matrice di input invece vengono sia passati n e m, numero di cellule e numero di mutazioni, che la matrice vera e propria rappresentata da un file di testo contente 1, 0, i quali indicano rispettivamente la presenza e la non presenza di una mutazione in una certa cellula, e 2, il quale indica che non ci sono abbastanza informazioni sulla presenza/assenza della mutazione in quella cellula (ciò che prima avevamo indicato con ?).
Possono poi essere passati i dati riguardanti il Simulated Annealing, ovvero la temperatura iniziale e il cooling rate, il quale indica quanto velocemente la temperatura diminuisce nel tempo.