UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Approcci algoritmici per scoprire le traiettorie conservate nelle filogenie tumorali

Relatore: Prof. Vandin Fabio

Laureando: Orsolon Ludovico

ANNO ACCADEMICO 2022 – 2023

Data di laurea 16/11/2023

Sommario

I tumori sono il risultato di un processo evolutivo che porta ad avere colonie di cellule contenenti differenze genetiche sostanziali. E' quindi di fondamentale importanza la ricerca di traiettorie conservate all'interno degli alberi filogenetici di diverse colonie tumorali; questa ricerca ci permette di conoscere meglio le mutazioni che culminano in tumori, così da poterne individuare e prevedere gli avanzamenti più velocemente. Uno strumento per far ciò è MASTRO che, a partire da un insieme di alberi filogenetici che descrivono una coorte di tumori, effettua anche un'analisi statistica delle traiettorie trovate, che gli permette di scartare quelle meno significative.

Questa tesi si focalizzerà sul problema di avere un numero diverso di alberi per ogni paziente, su un possibile modo per ovviare a questo problema e sull'analisi dell'efficacia della soluzione proposta, nel contesto dell'utilizzo di MASTRO. In particolare si verificherà l'efficacia di prendere un solo albero filogenetico (casuale) per ciascun paziente e se ne analizzeranno i vantaggi e gli svantaggi rispetto all'esecuzione di MASTRO sull'insieme di tutti gli alberi disponibili; si valuteranno i risultati ottenuti utilizzando un particolare dataset di filogenie di tumori ai polmoni. Si parlerà anche di un programma, sviluppato per questa tesi, in grado di convertire un determinato tipo di dataset (come quello menzionato) in un formato che MASTRO può analizzare.

Indice

Introduzione

Il processo di formazione dei tumori consiste nell'accumulazione di alterazioni somatiche che conferiscono vantaggi selettivi ad una, o ad un gruppo di cellule. L'accumulo di mutazioni rende il genotipo di un gruppo di cellule tumorali molto eterogeneo; ci sono però alcune caratteristiche, come l'ordine di apparizione di determinate mutazioni, che è comune in gran parte delle cellule di uno stesso gruppo tumorale. Trovare queste caratteristiche comuni ha una grande utilità, in quanto può aiutare la ricerca ad individuare tumori più velocemente e predirne l'avanzamento con più sicurezza.

Negli ultimi anni sono stati sviluppati diversi strumenti per far ciò, tra cui *MAximal tumor treeS TRajectOries (MASTRO)* [1], uno dei più recenti. MASTRO individua tutte le traiettorie comuni ad un gruppo di alberi filogenetici (dati come input) che descrivono l'evoluzione di un gruppo di tumori; è uno strumento particolarmente valido anche perché effettua un'analisi statistica sulle traiettorie trovate, scartando quelle che hanno un'alta probabilità di essere state individuate per caso e quindi di non essere significative. Il processo di sequenziamento del DNA che genera questi questi alberi tumorali però non è perfetto: per questo i dati che descrivono l'evoluzione dei tumori presentano quasi sempre incertezze, che si manifestano nell'avere un numero di alberi diversi per ogni paziente. MASTRO considera gli alberi che riceve in input come aventi tutti la stessa importanza, senza considerare a quale paziente in particolare appartengano; ne consegue che se un paziente ha molti più alberi di un altro, quest'ultimo avrà alberi di importanza minore rispetto quelli del primo, ciò non è ideale se si vuole fare un'analisi statistica.

In questa tesi si valuterà se la scelta casuale di un solo albero per ciascun paziente potrebbe alleviare questo problema; si faranno inoltre i confronti con i risultati ottenuti senza fare questa operazione, cercando di stimarne la bontà.

Concetti preliminari e definizioni per MASTRO[1]

MASTRO prende in input un insieme di alberi filogenetici che descrivono le evoluzioni di tumori di un gruppo di cellule, poi produce come output tutte le traiettorie osservate in almeno σ tumori, dove σ è una soglia impostata dall'utente al momento dell'esecuzione: se σ è impostata a 5 ad esempio, allora MASTRO scarterà tutte le traiettorie trovate che appaiono in meno di 5 degli alberi presenti come input; σ viene chiamata supporto minimo. Questi alberi descrivono le mutazioni subite da una cellula, non è assunto che queste mutazioni siano consecutive ma solo che avvengano nell'ordine indicato dallo specifico albero. Ogni albero tumorale $T = (V_t, E_t)$ è composto da un insieme di nodi V_t e un insieme di archi E_t. Ogni nodo V_t corrisponde ad un clone del tumore e contiene un insieme di mutazioni. La radice dell'albero rappresenta le cellule iniziali, che non hanno subito ancora nessuna mutazione (la *germline,* indicata con *g* o *GL*). Le mutazioni indicate in un nodo sono le mutazioni presenti in quel nodo ma non presenti nei suoi antenati, assumiamo che una mutazione possa avvenire una e una sola volta in ogni albero. Per ogni albero tumorale diciamo che una mutazione *a*, contenuta nel nodo *v*, è *un' antenata* della mutazione *b* contenuta nel nodo $w \neq v$, se il nodo *v* appartiene al percorso da *w* al nodo radice. Definiamo il supporto s_p di *P* in un dataset $D = \{T_1, \ldots, T_n\}$ come il numero di alberi appartenenti *D* dove *P* è osservata. Definiamo *una traiettoria P* come un albero tumorale $P = (V_p, E_p)$. Una traiettoria *P* è osservata in un albero *T* se il set di mutazioni contenute in *P* è un sottoinsieme delle mutazioni V_t in *T* e se ogni coppia V_{p1} e V_{p2} in *P* rispetta l'ordine temporale della coppia corrispondente V_{t1} e V_{t2} in *T*. MASTRO utilizza il *grafo espanso* ($G_T = (V_T{}^G, E_T{}^G)$) che è un grafo diretto che si ottiene a partire dall'albero filogenetico di un tumore, applicando ad esso i seguenti passaggi:

[1] Queste informazioni sono contenute nel paper [1]

1. Dividere i nodi che contengono più di una mutazione in nodi separati
2. Collegare i nodi separati nel punto 1 con un arco speciale, denotato dal simbolo ⋆, questo arco indica che non conosciamo l'ordine esatto di arrivo delle mutazioni
3. Inserire archi in modo che ogni nodo sia collegato con ciascuno dei suoi discendenti

Dato un Grafo G è facile osservare che una traiettoria P è presente in un albero T se e solo se il grafo G_P è un sottografo di G_T. Indichiamo una traiettoria P come *massima* se, aggiungendo una qualsiasi mutazione all'albero analizzato o a P, il supporto s_p diminuisce. Lo scopo finale di MASTRO è quindi scoprire *traiettorie massime frequenti*.

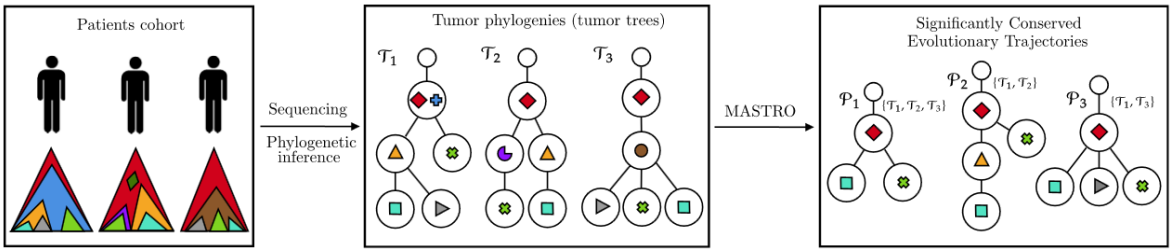


Fig. 1. Una descrizione ad alto livello del funzionamento di MASTRO. Viene mostrato come si ottengono (tramite il sequenziamento del DNA) le filogenie tumorali strutturate come indicato precedentemente: diverse mutazioni corrispondono a diverse forme colorate indicate nel secondo riquadro. Da questi dati iniziali MASTRO è in grado di trovare delle traiettorie, indicate nel terzo riquadro come P_1,P_2 e P_3. I supporti di queste traiettorie sono indicati tra parentesi quadre.
Figura presa da [1].

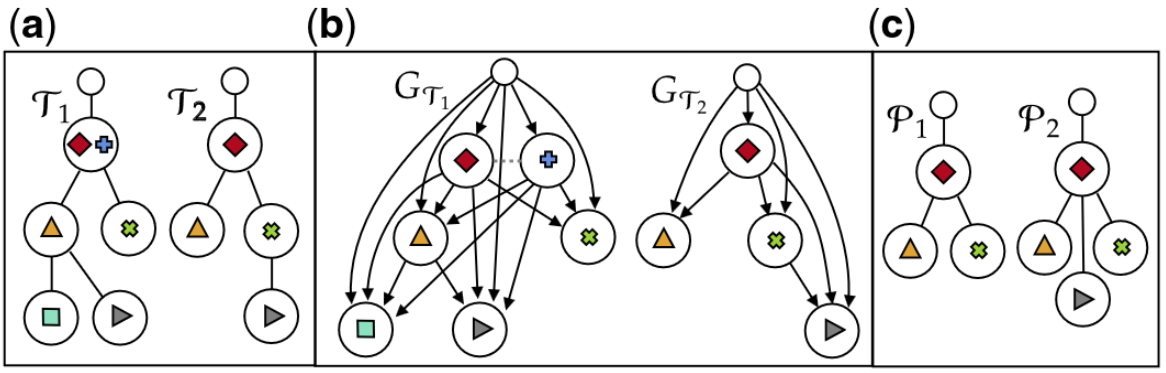


Fig. 2. (a) Alberi tumorali di T_1 e T_2. **(b)** I grafi tumorali espansi G_{T1} e G_{T2} (i trattini sono il collegamento denotato con ⋆) **(c)** le traiettorie trovate come sottografi di G_{T1} e G_{T2}.
Figura presa da [1].

Formato di input per MASTRO

Nella pratica, l'elaborazione di MASTRO viene fatta a partire da un documento di testo nel quale, su ogni riga, è espresso l'insieme di archi del grafo espanso di una filogenia tumorale. Ogni riga del file è quindi formata da una serie di coppie di nodi collegati in tre modi diversi a seconda del tipo di arco:

- *A->-B* indica che *A* è un antenato di *B*
- *A-/-B* indica che *A* e *B* appartengono a rami diversi e che quindi nessuno dei due è antenato dell'altro
- *A-?-B* indica che non conosciamo l'ordine tra *A* e *B*

Due nodi collegati da archi del tipo "-?-" e "-/-" devono essere ordinati in ordine lessicografico (quello che viene prima deve stare a sinistra del collegamento).
Come accennato in precedenza questo metodo non permette di fare distinzione tra grafi appartenenti a diversi pazienti, in quanto ogni riga (e quindi ogni grafo) è indipendente dalle altre.

Dataset utilizzati

L'analisi di questa tesi tramite MASTRO utilizzerà soprattutto il dataset *tracerx_lung*[2], utilizzato dal gruppo di ricerca del prof. *El-kebir* per la validazione di uno strumento simile a MASTRO e precedente a esso: RECAP [2].
tracerx_lung è un dataset che contiene 99 pazienti e 137 alberi i cui grafi espansi, in alcuni casi, sono di notevole lunghezza (si parla di anche 50 archi a grafo). Nell'analizzare *tracerx_lung* la notevole lunghezza di alcuni dei grafi diventerà un problema, in quanto causa un allungamento notevole del tempo di esecuzione di MASTRO rispetto a dataset più "semplici", rendendo anche oneroso il carico di risorse necessarie per questa esecuzione. Per risolvere questo problema e permettere l'agevole confronto dei dati, in alcune istanze *tracerx_lung* verrà convertito in *tracerx_lung_less_than_26;* questo "nuovo" dataset conterrà solo i grafi di *tracerx_lung* che hanno un numero di archi minore o uguale a 26, escludendo quindi quelli con un numero maggiore. 26 è un valore selezionato empiricamente che permette di scartare meno grafi possibili da *tracerx_lung,* permettendo al contempo di mantenere contenuti i tempi di esecuzione. Per alcuni esempi e

[2] https://github.com/elkebir-group/RECAP/blob/master/data/TRACERx_lung/tracerx_lung.txt

confronti verranno utilizzati anche altri due dataset: *treeforpars* e *breast_Razavi*[3]. Il primo è un dataset molto piccolo (3 pazienti e 4 alberi) creato solo per fare test e spiegare esempi, il secondo invece è un dataset molto grande (1315 pazienti e 37809 alberi), usato anch'esso per la validazione di RECAP.

Il formato di input di RECAP è diverso da quello di MASTRO in modo fondamentale: RECAP distingue gli alberi appartenenti a pazienti diversi e inoltre non riceve come input il grafo espanso, ma una descrizione dell'albero in sé.

In particolare un file di input di RECAP è formato in questo modo:

- Ogni riga che non inizia con un numero rappresenta un arco dell'albero; l'arco viene indicato con il nome del padre seguito da uno spazio seguito dal nome del figlio
- Se un nodo ha dentro di sé più mutazioni (perchè non si può determinarne l'ordine) queste sono separate da un ","
- Sono presenti delle righe che indicano il numero di alberi per ogni paziente e delle altre che indicano il numero di archi per ogni albero
- È presente una riga iniziale che indica il numero totale di pazienti

Se si vuole effettuare un'analisi con MASTRO di dati creati per RECAP diventa necessario quindi uno strumento che possa effettuare la conversione di questi nel formato accettato da MASTRO. Tale strumento è stato sviluppato per questa tesi ed è descritto nel capitolo successivo.

[3] https://github.com/elkebir-group/RECAP/blob/master/data/breast_Razavi/breast_Razavi.txt

Un parser per l'input e l'output di MASTRO

In questo capitolo verrà analizzato lo strumento sviluppato per questa tesi che permette di convertire dataset espressi per il funzionamento con RECAP in dataset utilizzabili con MASTRO.
I dettagli su come eseguire il programma sono presenti nel README alla pagina GitHub di *ParserForMastro*[4] citata in questa tesi. Tutti i file menzionati e i risultati delle analisi riportate sono presenti nella stessa pagina GitHub, le informazioni sul computer dove i test sono stati eseguiti sono riportate nell'ultimo capitolo di questa tesi.
Il programma è realizzato interamente nella versione 3 di Python ed è stato realizzato per essere compatibile con linux.

Librerie utilizzate[5]

argparse: utilizzata per consentire il passaggio di parametri su terminale da esecuzione.

random: utilizzata per selezionare casualmente un albero tra quelli di un paziente.

os: utilizzata per poter lanciare comandi del terminale da dentro il programma.

networkx: utilizzata per la creazione,la manipolazione e la gestione di grafi.

matplotlib: utilizzata per la visualizzazione opzionale dei grafici degli alberi da convertire.

[4] https://github.com/OrsolonLudovico/ParserForMastro
[5] I collegamenti alle risorse riguardanti le varie librerie sono riportati nella sezione "*Collegamenti e risorse utilizzate*".

Composizione del programma

ParserForMastro è composto principalmente da *parserDataMASTRO.py* e *run_parser.py*. *parserDataMASTRO.py* racchiude la funzione che permette di convertire un albero dal formato RECAP al grafo necessario per MASTRO.
run_parser.py invece serve come funzione chiamante che legge gli alberi da un file, li passa a *parserDataMASTRO.py* e salva il risultato in un file di output, questo script implementa anche delle opzioni aggiuntive:

- È possibile specificare il numero massimo di archi che un grafo espanso può avere, impostando 26 sul dataset *tracerx_lung.txt* si ottiene "*tracerx_lung_converted_**all**_less_than_26.txt"* di cui si è parlato in precedenza assieme alla sua versione con un solo albero per paziente: "*tracerx_lung_converted_**rand**_less_than_26.txt".* A volte, per brevità, i dataset di tipo *all* saranno chiamati semplicemente *dataset all* e quelli di tipo rand *dataset rand*.
- È possibile decidere se ottenere l'output anche su terminale oltre che su file
- È possibile decidere se convertire tutti gli alberi trovati nell'input, solo uno per paziente o fare entrambe le cose. Se si deciderà di limitare il numero di archi nel grafo espanso, quando il programma dovrà scegliere un singolo albero per paziente, sceglierà un albero casuale di quel paziente tra quelli che rispettano la condizione impostata. Nel caso specifico del dataset *tracerx_lung* se un paziente ha un grafo espanso di lunghezza maggiore di 26 allora anche tutti gli altri grafi di quel paziente avranno lunghezza maggiore di 26: questo risulta nell'eliminazione totale del paziente dal dataset sia dalla versione *all* che dalla versione *rand* .

Esempi di funzionamento

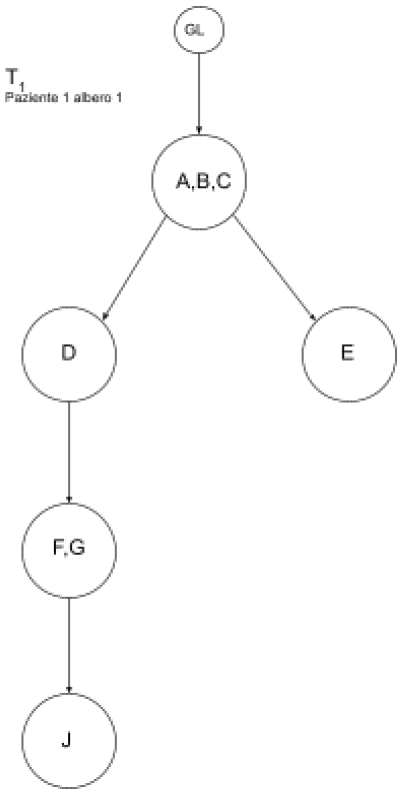
In questo paragrafo verranno analizzati alcuni esempi di funzionamento di *ParserForMastro*, ne verrà descritto l'input e verificato l'output.

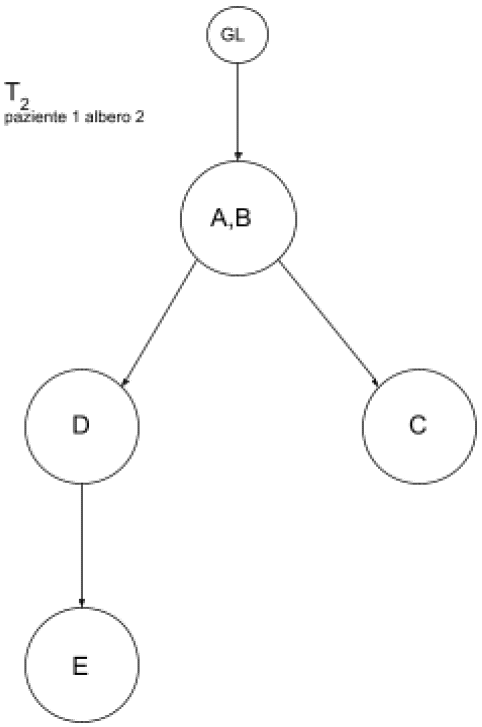
Il dataset che verrà utilizzato per fare queste operazioni sarà *treeforpars*, descritto in precedenza, esso è stato costruito proprio con lo scopo di essere più esemplificativo possibile.

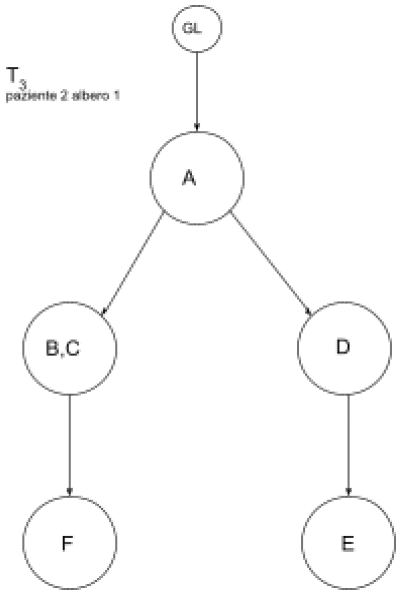
Il contenuto di *treeforparse* è il seguente:

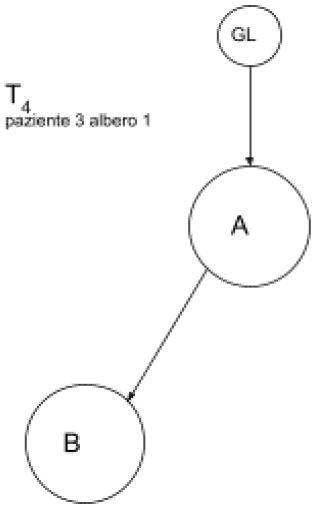
```
3 #patients
2 #trees for patient 1
5 #edges
GL C;B;A
A;C;B E
A;B;C D
D F;G
F;G J
4 #edges
GL A;B
B;A D
A;B C
D E
1 #trees for patient 2
5 #edges
GL A
A B;C
A D
B;C F
D E
1 #trees for patient 3
2 #edges
GL A
A B
```

Questi dati codificano per i seguenti alberi:









Fornendo al programma questo dataset ed eseguendo il comando:

```
python3 run_parser.py -g treeforpars.txt
```

[-g indica che la stringa seguente è il file di input che vogliamo utilizzare]

Otteniamo due file:

- *Treeforpars_converted_**all**.txt*: contiene tutti gli alberi trovati convertiti nel formato accettato da MASTRO, il suo contenuto è:

```
A-?-C B-?-C A-?-B F-?-G D-/-E E-/-F E-/-G E-/-J C->-D
C->-J C->-F C->-G C->-E B->-D B->-J B->-F B->-G B->-E
A->-D A->-J A->-F A->-G A->-E D->-J D->-G D->-F F->-J
G->-J \n
A-?-B C-/-D C-/-E A->-E A->-D A->-C B->-E B->-D B->-C
D->-E \n
B-?-C B-/-D B-/-E C-/-D C-/-E D-/-F E-/-F A->-D A->-F
A->-C A->-E A->-B B->-F C->-F D->-E \n
A->-B \n
```

[per evitare confusione è stato aggiunto un "\n" manualmente alla fine di ogni grafo per far notare dove è il file ad andare a capo e dove si va a capo per questioni di spazio]

Come previsto, ogni riga rappresenta correttamente l'albero corrispondente fornito come input

- *Treeforpars_converted_**rand**.txt:* contiene solo un albero per ogni paziente scelto casualmente, il suo contenuto è:

```
A-?-C B-?-C A-?-B F-?-G D-/-E E-/-F E-/-G E-/-J C->-D
C->-J C->-F C->-G C->-E B->-D B->-J B->-F B->-G B->-E
A->-D A->-J A->-F A->-G A->-E D->-J D->-G D->-F F->-J
G->-J \n
B-?-C B-/-D B-/-E C-/-D C-/-E D-/-F E-/-F A->-D A->-F
A->-C A->-E A->-B B->-F C->-F D->-E \n
A->-B \n
```

  Essendo stato selezionato un solo albero per paziente, i 3 pazienti analizzati corrispondono a 3 alberi di output. Con un'altra esecuzione l'albero scelto per il paziente numero uno potrebbe variare, mentre quelli scelti per i pazienti 2 e 3 non varieranno mai, in quanto questi pazienti hanno solo un albero ciascuno.

Convertendo invece il dataset con questa riga di codice:

```
python3 run_parser.py -g treeforpars.txt -s 26
```

La flag *-s* indica che il numero inserito è il massimo numero di archi che può avere il grafo espanso di ogni albero convertito. Otteniamo quindi gli stessi due file di prima, questa volta però senza il primo albero del primo paziente, questo albero infatti, una volta convertito, avrebbe avuto più di 26 archi nel suo grafo.
Visto che il primo paziente aveva solo 2 alberi e uno di questi è stato cancellato, ora ogni paziente ha un solo albero, ciò significa che in questo caso i due output sono identici:

```
A-?-B C-/-D C-/-E A->-C A->-E A->-D B->-C B->-E B->-D D->-E
/n
B-?-C B-/-D B-/-E C-/-D C-/-E D-/-F E-/-F A->-E A->-D A->-F
A->-B A->-C B->-F C->-F D->-E /n
A->-B /n
```

Qui riportati *tracerx_lung_converted_**all**_less_than_26* e *tracerx_lung_converted_**rand**_less_than_26* che in questo caso sono uguali

Tempi di esecuzione

Utilizzando i tre dataset descritti nella sezione "*Dataset utilizzati*" per l'esecuzione di *ParserForMastro,* i tempi medi di esecuzione sono riportati nella tabella seguente:

	treeforpars	tracerx_lung	breasts_Razavi
tempo medio esec.	0.349 s	0.360 s	24.348 s
N° alberi	4	137	37809
tempo / n° alberi	$8.73 \cdot 10^{-2}$ s	$2.63 \cdot 10^{-3}$ s	$6.44 \cdot 10^{-4}$ s

Tabella 1.

Notiamo che, con l'aumentare dei campioni, il tempo di esecuzione in rapporto al numero di alberi diminuisce, ciò significa che il tempo di elaborazione di un singolo albero si abbassa; la causa di ciò è probabilmente da attribuire al fatto che è presente un fattore di tempo costante nell'esecuzione del programma (come il tempo di aprire e chiudere il file di output). Con l'aumentare degli alberi, e quindi del tempo di esecuzione totale del programma, questo tempo iniziale costante (dovendo essere diviso per il numero di alberi) ha sempre meno impatto sul tempo di esecuzione di un singolo albero; questo fa sì che il tempo di esecuzione del singolo albero in *ParserForMastro* sia inversamente proporzionale al numero di alberi totali.

Da questi risultati si evince anche che il tempo di esecuzione non aumenta esponenzialmente con l'aumentare dei dati, si può quindi pensare che il tempo di esecuzione di *ParserForMastro* abbia andamento lineare.

Utilizzo e confronto dei dati con MASTRO

In questo capitolo gli output di *ParserForMastro* verranno analizzati da MASTRO, si cercherà di capire che cosa comporta la scelta randomica di un albero per ogni paziente e se ne descriveranno vantaggi e svantaggi rispetto all'analisi indistinta di tutti gli alberi, concentrandosi principalmente sul dataset *tracerx_lung*.

Validità del campionamento randomico

Un primo test per determinare la validità della scelta casuale di un albero per paziente è verificare quanto i dataset generati con questo processo casuale sono diversi tra di loro. Se utilizziamo *run_parser.py* per convertire i dataset di *tracerx_lung*, i file "*rand"* (quelli contenenti un solo albero per ogni paziente) sono diversi tra di loro per ogni esecuzione: differenza ovviamente dovuta alla natura casuale della scelta degli alberi che genera questi file. Dando però in pasto questi differenti file *rand* a MASTRO, notiamo che i vari output di quest'ultimo sono quasi del tutto uguali: il numero di traiettorie trovate nei diversi file di input è praticamente identico. Nel caso specifico di *tracerx_lung* le traiettorie trovate in 10 versioni diverse di *tracerx_lung_converted_**rand**.txt* sono sempre 93 o 94, una variazione di poco più dell'1%. Facendo lo stesso confronto con dataset contenenti ancora meno alberi, la differenza di numero addirittura scompare; anche le traiettorie trovate usando diversi campioni sono praticamente identiche. È ragionevole pensare che la differenza nel numero di traiettorie trovate diminuisca con l'aumentare dell'eterogeneità degli alberi di ciascun paziente; infatti, nel dataset *tracerx_lung*, gli alberi di ciascun paziente sono molto simili tra di loro. Questa similitudine porta al ritrovamento di alcune traiettorie che sono presenti solo tra gli alberi di singoli pazienti, che ovviamente non possono essere trovate nella versione *rand,* traiettorie che fanno aumentare la differenza tra il numero di traiettorie della versione *all* e il numero di traiettorie nella versione *rand*. È inoltre importante precisare che, in un qualsiasi dataset che rappresenti dati reali, i vari alberi di ciascun paziente non possono essere eccessivamente diversi: l'avere più alberi per ogni paziente è il risultato dell'incertezza nel sequenziamento, se questi alberi fossero molto diversi significherebbe che il processo di sequenziamento è incorretto.

Considerazioni sul supporto minimo

Nell'eseguire MASTRO, come già accennato, è possibile specificare il supporto minimo che una traiettoria deve avere per essere considerata: specificare un numero alto velocizza notevolmente l'esecuzione. Un numero alto del supporto minimo rischia anche di aumentare il bias dovuto alla similarità tra gli alberi di un singolo paziente, ma questo aspetto verrà analizzato in una parte successiva della tesi.

Mentre l'esecuzione di MASTRO su *tracerx_lung_converted_**all*** è molto lunga e onerosa in termini di risorse, l'esecuzione su *tracerx_converted_**rand*** è invece relativamente veloce; viene da chiedersi il perchè, nonostante la versione *rand* contenga comunque grafi con quantità elevate di archi, questa sia così leggera. La risposta si trova nel supporto minimo (di default MASTRO considera questo valore 2). Aumentando manualmente questo valore a 14 ecco che l'esecuzione con *tracerx_lung_**all*** diventa veloce e leggera, quanto quella con la sua versione *rand*. Tale aspetto è dovuto al fatto che, nel dataset preso in considerazione, i pazienti con gli alberi più grandi hanno un massimo di proprio 14 alberi a testa; MASTRO quindi "scarta" l'analisi di questi alberi evitando la parte più onerosa che causava rallentamenti. Ma *tracerx_lung_**rand*** contiene anch'esso alberi complessi, eppure la sua esecuzione non è onerosa. La causa di questo fenomeno è da ritrovare nel fatto che gli alberi contenuti nella versione *rand*, anche se sono grandi quanto quelli della versione *all,* sono più diversi tra loro; basta che il supporto minimo sia uguale al valore di default (2) per scartare vari alberi particolarmente complessi e diversi dalla maggioranza di tutti gli altri.

Tempi di esecuzione di MASTRO al variare del numero minimo di supporti

In questa sezione verranno riportati e confrontati i tempi medi di esecuzione di MASTRO utilizzando come input i dataset *tracerx_lung_converted_**all**_less_than_15* e *breast_Razavi_converted_**all**_less_than_15* assieme alle loro versioni *rand*, in cui è presente un solo albero per paziente. Questi dataset sono stati ottenuti limitando il numero massimo di archi nei grafi ottenuti da *tracerx_lung* e *breast_Razavi* a 15.

	tracerx_lung < 15		breast_Razavi < 15	
Supporto minimo ⬇	ALL	RAND	ALL	RAND
2	1.120 s	0.744 s	101.132 s	23.141 s
5	0.541 s	0.537 s	26.151 s	9.027 s
8	0.5165 s	0.514 s	14.388 s	5.736 s

Tabella 2.

Si nota subito che con il numero minimo di supporti più basso, il tempo risparmiato eseguendo il dataset *rand* rispetto alla versione *all* è, in proporzione, maggiore. Non essendo presente (volutamente) in questi due dataset nessun albero molto complesso (il numero massimo di archi del grafo è limitato a 15) si può concludere che il motivo di questo risparmio di tempo sia da imputare al numero di alberi presenti nei dataset *rand,* che è significativamente minore del numero di alberi contenuti nella sua controparte *all*. *breast_Razavi* ha molti più alberi per paziente rispetto a *tracerx_lung,* infatti si osserva che il tempo risparmiato con la sua versione *rand* è maggiore in proporzione a quello risparmiato con la versione *rand* di *tracerx_lung,* ciò conferma l'ipotesi per cui il tempo guadagnato dall'esecuzione della versione *rand* rispetto alla versione *all* è proporzionale alla percentuale di alberi in meno che ha la versione *rand* rispetto alla *all*.

Confronti in dati TRACERx

In questo paragrafo verranno confrontati gli output di MASTRO utilizzando come input *tracerx_lung* nella sua versione con tutti gli alberi (*all*) e nella sua versione con un albero per ogni paziente (*rand*), verranno fatte anche delle considerazioni utilizzando il supporto minimo variabile. Entrambe le versioni sono state filtrate come descritto nel funzionamento di *ParserForMastro* per non avere grafi con più di 26 archi. Nella tabella sono espressi i numeri e le lunghezze medie delle traiettorie trovate per la versione *rand* e la versione *all* di *tracerx_lung*.

	Numero traiettorie trovate			Lunghezza media traiettorie		
Supporto Minimo ⬇	ALL	RAND	%	ALL	RAND	%
2	122	93	- 23.77 %	8.4	5.5	- 34.52 %
5	24	15	- 37.50 %	8.64	3.9	- 54.86 %
8	11	5	- 54.54 %	7.6	2.83	- 62.76 %
9	8	4	- 50 %	9	2.8	- 68.89 %
10	8	4	- 50 %	9	2.8	- 68.89 %
11	8	4	- 50 %	9	2.8	- 68.89 %
14	3	2	- 33.33 %	2.75	2.3	- 16.36 %

Tabella 3. La quarta colonna indica la percentuale di traiettorie trovate in meno nella versione *rand* rispetto alla versione *all* (Es: 93 è il 23.77% in meno di 122). L'ultima colonna indica la percentuale di lunghezza in meno delle traiettorie trovate nella versione *rand* rispetto alla versione *all* (Es: 5.5 è il 34.52 in meno di 8.4)

Se osserviamo i risultati ottenuti notiamo che (almeno fino a che il supporto minimo non supera 8) mano a mano che aumentiamo il supporto minimo, la differenza percentuale nel numero di traiettorie individuate da MASTRO nelle differenti versioni aumenta. Molti dei pazienti in *tracerx_lung* hanno vari alberi molto simili tra loro, nell'output di MASTRO saranno quindi presenti alcune traiettorie che si trovano solo negli alberi che appartengono ad un singolo paziente. La maggior parte di queste traiettorie avranno supporto pari al numero di alberi del paziente a cui appartengono; ciò comporta che esse possono essere trovate con qualsiasi supporto minimo inferiore al numero di alberi del dato paziente. Essendo il dataset *rand*, composto solo da alberi di diversi pazienti, esso non possiede

questo tipo di traiettorie "*intra-paziente*". Quindi, se il supporto minimo aumenta, *rand* perde in percentuale più traiettorie di quante ne perda *all: rand* infatti non può contare sulle traiettorie che vengono trovate indipendentemente dal supporto minimo come può fare *all*. Impostando il supporto minimo a 9 o più, avviene proprio il superamento da parte di questo del numero di alberi appartenenti ad un particolare paziente, il quale possiede alberi dai grafi particolarmente lunghi e omogenei. Se il supporto minimo supera 11 invece, questo diventa maggiore del numero di alberi di un ulteriore paziente, facendo variare le differenze e collassare il grafico. Alzare il supporto minimo causa quindi un aumento del bias dovuto a pazienti con alberi numerosi e omogenei; il problema non sussiste nel momento in cui si utilizza la versione *rand,* che non possiede più di un albero per paziente.

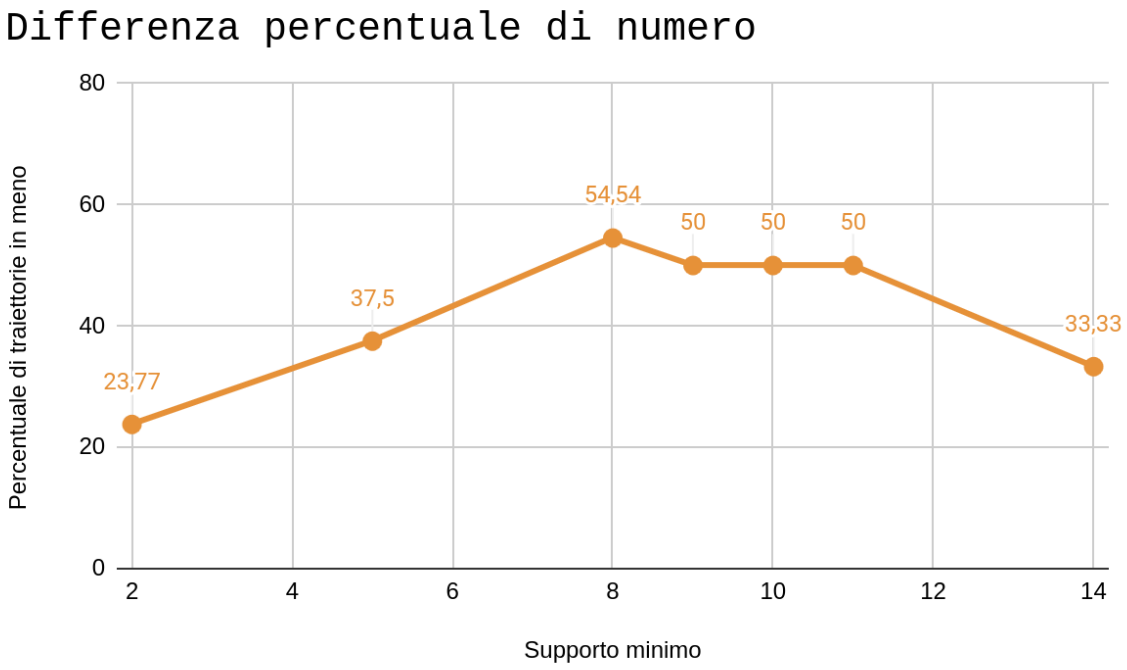


Grafico 1. Vengono rappresentate le coppie "percentuale in meno di traiettorie trovate su rand rispetto ad *all*" e "supporto minimo"

Nella versione *rand*, più il supporto minimo cresce più corte saranno le traiettorie trovate (naturalmente è più probabile che una traiettoria corta sia comune ad un numero maggiore di grafi rispetto ad una lunga). Nel dataset *all* invece, le cose sono diverse a causa della similarità delle traiettorie che appartengono allo stesso paziente. Le traiettorie trovate tra gli alberi di singoli pazienti infatti, hanno lunghezze elevate proprio a causa di questa similarità; se aumentiamo il supporto minimo le traiettorie trovate tra singoli pazienti occupano una maggior percentuale delle traiettorie totali, facendo si che la lunghezza media delle traiettorie trovate non diminuisca ma aumenti (almeno fino a che il supporto

minimo non superi il numero di alberi appartenenti ai pazienti con gli alberi più grandi). Se la lunghezza media degli alberi di *all* cresce mentre quella degli alberi di *rand* decresce viene da sé che la differenza, in proporzione, tra le lunghezze medie di *all* e *rand* aumenti. C'è da aggiungere che quando il supporto minimo raggiunge 14, questo supera il numero di alberi di ogni singolo paziente nel dataset eliminando la correlazione osservata in precedenza

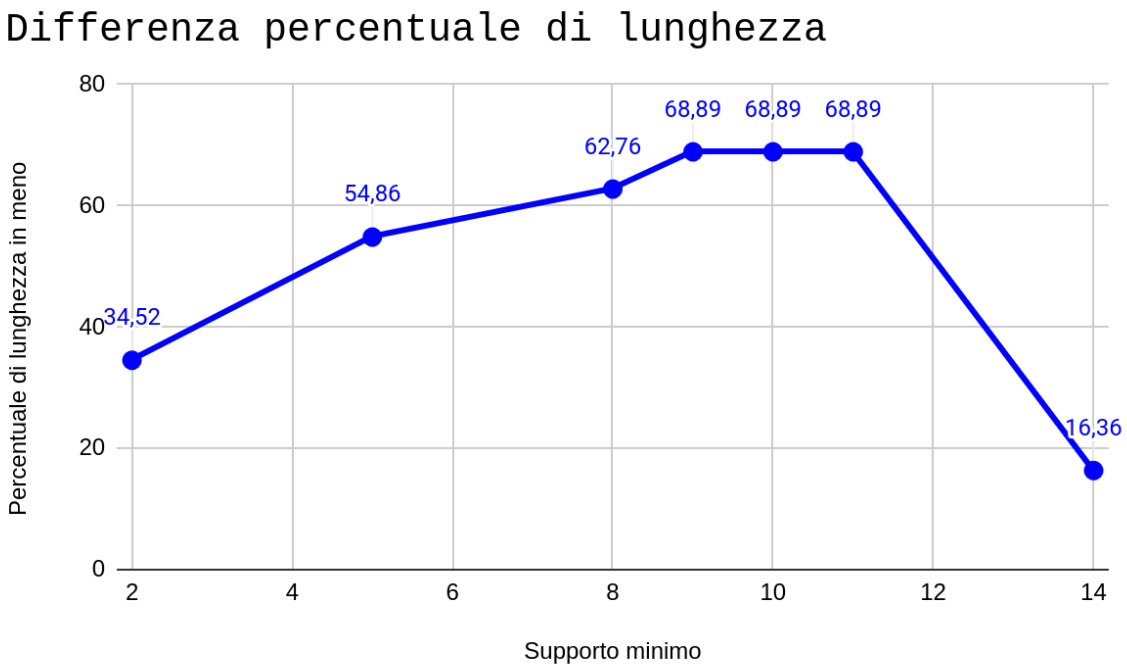


Grafico 2. Vengono rappresentate le coppie "Percentuale di lunghezza in meno di *rand* rispetto ad *all*" e "supporto minimo"

Conclusioni

Il quesito posto all'inizio della tesi chiedeva quali fossero i vantaggi e gli svantaggi di prendere un solo albero per ogni paziente al posto di tutti gli alberi di un dataset.
Un primo vantaggio, all'apparenza banale ma che in realtà è di fondamentale importanza, è la drastica diminuzione del tempo, e delle risorse, necessari per l'esecuzione di MASTRO. Utilizzare un dataset con un solo albero per paziente può significare la differenza tra poter eseguire MASTRO e non avere le risorse per farlo. Inoltre questo metodo rende anche molto più pratiche esecuzioni consecutive di MASTRO per effettuare studi e confronti. Un secondo punto di forza di selezionare un solo albero per paziente è che si eliminano dall'output finale tutte quelle traiettorie che sono ritrovate solo internamente agli alberi di un solo paziente e che quindi non hanno un grande significato statistico.
Anche con questi pregi, l'analisi di un singolo albero per paziente non può evidentemente sostituire l'analisi della totalità degli alberi: selezionando un solo albero per paziente c'è un'inevitabile perdita di dati (riducibile generando e analizzando più volte i dataset randomici) che rende questa tecnica non ottimale per alcuni casi d'uso.

Collegamenti e risorse utilizzate

In questa ultima sezione verranno riportate le fonti e le risorse su cui si è basata questa tesi:

MASTRO

La documentazione e le risorse riguardanti MASTRO possono essere trovate alla pagina GitHub dedicata:
https://github.com/VandinLab/MASTRO
MASTRO è uno strumento realizzato dal Vandin Lab la cui pagina è presente all'indirizzo:
https://www.dei.unipd.it/~vandinfa/

RECAP

I dataset utilizzati sono stati usati per la realizzazione di RECAP e si trovano alla pagina:
https://github.com/elkebir-group/RECAP/tree/master/data

ParserForMastro

Informazioni riguardo ParserForMastro e risultati ottenuti con esso si trovano alla pagina GitHub realizzata per questa tesi a lui dedicata:
https://github.com/OrsolonLudovico/ParserForMastro
Le librerie utilizzate per la creazione di ParserForMastro possono essere trovate ai seguenti link:
argparse: https://docs.python.org/3/library/argparse.html
random: https://docs.python.org/3/library/random.html
os: https://docs.python.org/3/library/os.html
networkx: https://networkx.org/documentation/stable/tutorial.html
matplotlib: https://matplotlib.org/stable/index.html

Bibliografia

[1] Leonardo Pellegrina, Fabio Vandin, Discovering significant evolutionary trajectories in cancer phylogenies, *Bioinformatics*, Volume 38, Issue Supplement_2, September 2022, Pages ii49–ii55, https://doi.org/10.1093/bioinformatics/btac467

[2] Christensen S, Kim J, Chia N, Koyejo O, El-Kebir M. Detecting evolutionary patterns of cancers using consensus trees. Bioinformatics. 2020 Dec 30;36(Suppl_2), https://pubmed.ncbi.nlm.nih.gov/33381820

Macchina utilizzata

Il computer utilizzato per la creazione delle tabelle e per la misurazione dei tempi:
Sistema operativo Fedora Linux
Archiviazione 140 MB/s read 80 MB/s write
CPU AMD desktop 8 core 16 thread, 4.2 GHz di clock durante i test.
RAM 3600 MHz

Un ringraziamento al professor Vandin Fabio che mi ha seguito nella realizzazione di questa tesi e alla mia famiglia che mi ha sostenuto per tutta la mia carriera universitaria.